Escuela Politécnica Superior

EnableParking

Máster Universitario en Desarrollo de Software para Dispositivos Móviles

Trabajo Fin de Máster

Autor:
José Manuel Bru Sempere

Tutor/es:
Miguel Ángel Lozano Ortega

Septiembre 2019

Universitat d'Alacant
Universidad de Alicante

Universidad de alicante

Escuela Politécnica Superior

EnableParking

Diseño y programación de una aplicación móvil de Innovación Social Digital

Autor
José Manuel Bru Sempere

Tutor
Miguel Ángel Lozano Ortega

Resumen

La tecnología y los procesos informáticos son más motivadores si cabe cuando los enfocamos a la mejora de procesos que de otra forma pueden ser más complicados, más aún si podemos utilizar esta tecnología para ayudar a personas cuyos procedimientos a seguir en su vida diaria no son demasiado intuitivos o pueden llegar a ser complicados de forma que lleguen a condicionar sus vidas de alguna forma.

Por eso surge EnableParking, con el objetivo de ayudar y volver un poco más sencillo el proceso que siguen las personas con problemas de movilidad a la hora de desplazarse, puesto que para estas personas requiere un esfuerzo mayor de lo que muchos pensaríamos y una organización y previsión por encima de lo que los demás hacemos en nuestro día a día.

Con el desarrollo e implantación de esta idea que ha surgido, tanto por parte del grupo Artefactos de la Universidad de Alicante como de la Asociación para la integración sociolaboral de personas con discapacidad física y sensorial (AMFI), y con lo que yo he podido aportar, ayudamos a que la planificación a la hora de movilizarse y buscar una plaza de aparcamiento les sea un poco más sencilla.

Mediante el uso de EnableParking podremos saber dónde encontrar plazas públicas dirigidas a este grupo de personas, ayudarles a llegar mediante GPS y una interfaz sencilla y además, tener información sobre cuáles están libres y cuáles no.

Agradecimientos

A mi familia, por ser un apoyo siempre que es necesario y ayudar en todos los ámbitos de mi vida.

A mi tutor, por guiarme a lo largo de estos meses y aconsejarme durante todo el proyecto siempre que lo he necesitado, incluso en días festivos.

A mis compañeros de Artefactos por ser la mitad de este proyecto y estar siempre pendiente de mis dudas.

A mis amigos por darme ánimos, porque saben lo que es estudiar y trabajar al mismo tiempo y me han ayudado a tirar adelante.

Y por último a mi pareja, porque ha aguantado todas las horas pegado al ordenador aun cuando el humor no era mi mejor aliado.

Gracias.

Tabla de contenido

Índice de figuras

Índice de Tablas

Lista de acrónimos

MVC: Modelo-Vista-Controlador

UC: Use case / Caso de uso

DB: Base de datos

ERS: Especificación de requisito software

RF: Requisito funcional

RNF: Requisito no funcional

AMFI: Asociación para la Integraciónsociolaboral de Personas con Discapacidad Física y Sensorial

IEEE: Institute of Electrical and Electronics Engineers

API: Application Programming Interface

TIC: Tecnologías de la Información y la Comunicación

INE: Instituto Nacional de Estadística

SDK: Software Development Kit

UA: Universidad de Alicante

1. Introducción y justificación del proyecto

No podemos apartar la mirada y hacer como que las personas discapacitadas no existen. Un pensamiento que he tenido a raíz de formar parte de este proyecto es:

Que las personas discapacitadas pasen mucho tiempo en casa y les sea complicado el movilizarse para hacer su vida solo significa que debemos ayudarles a que salir a la calle sea una tarea más sencilla. Y la forma de hacerlo es facilitándoles la vida mediante las herramientas que tengamos a nuestro alcance.

Y es que la cantidad de personas que poseen algún tipo de discapacidad, solo en España, es muy alta. Un censo social oficial de hace unos 10 años, encuesta nombrada "**Discapacidades, autonomía personal y situaciones de dependencia**" [1], ya señalaba que el número de españoles con discapacidad rondaba los casi 4 millones, donde si filtramos los datos podemos ver como las discapacidades de tipo movilidad estaban presentes en la vida de alrededor de 2,54 millones de personas, con grandes necesidades.

En nuestra vida cotidiana vemos a menudo en centros comerciales, universidades e incluso en las calles de nuestras casas, plazas destinadas a estas personas. Ya sean de carácter personal (las cuales son solicitadas a la administración pública) como de carácter público, en las que cualquier persona con un documento legal que lo acredite puede hacer uso de ellas.

Este tipo de plazas han surgido como solución a la problemática que tienen estas personas a la hora de desplazarse a lugares tan cotidianos como ir a hacer la compra para su casa, de forma que estas plazas suelen estar cerca de las entradas y salidas de los lugares de interés, también denominados centros de actividad (determinada esta condición por los ayuntamientos).

Además, habitualmente poseen unas medidas más grandes a las de una plaza normal, pues esto les ayuda a desplegar material adicional a la hora de subir y bajar del vehículo si fuese necesario (ej. una silla de ruedas, rampas…).

Este proyecto, en el que nos encontramos, surge desde la **Asociación para la integración sociolaboral de personas con discapacidad física y sensorial** (AMFI [2]) y el grupo Artefactos de la Universidad de Alicante (UA), que nos han hecho llegar una problemática que sufren desde hace mucho tiempo en las plazas de aparcamiento antes introducidas y que de momento no han podido solventar.

El principal problema es el estacionamiento indebido por parte de los ciudadanos en plazas de aparcamiento dirigidas a personas discapacitadas, ya sean privadas o públicas. De tal forma que este colectivo, además de la problemática y el gasto de recursos que les supone un desplazamiento, que a priori sería completamente usual para una persona sin esta condición, también se le hace perder el tiempo con la consecuente posible pérdida de recursos por parte del afectado. Dado que deben avisar personalmente a las autoridades una vez se ven envueltos en la situación.

Además, este tema es más usual de lo que pudiéramos pensar en un principio. Haciendo una simple búsqueda podemos encontrar cantidad de artículos y blogs hablando al respecto [3].

Podemos leer relatos de personas afectadas en su vida diaria y cuentan cómo la gente suele ser bastante irrespetuosa con las plazas.

Existiendo casos en los que incluso el infractor toma represalias o bien contra la persona con discapacidad o bien contra el vehículo de este cuando las autoridades toman cartas en el asunto o simplemente cuando se les intenta concienciar de que el comportamiento que han tenido no es el adecuado. De forma que las personas desfavorecidas nunca saben cómo va a reaccionar la persona que incumple las normas, lo que condiciona la forma de actuar de estas personas, aunque tenga la razón en la disputa.

Pero no solo esto puede llegar a ser solucionable, si no que existen problemas de una índole menor como pueda ser la falta de información y localización de estas plazas en muchas ciudades. De forma que si alguien viaja o visita lugares (por placer, vacaciones u otro motivo) puede tener un problema a la hora de saber dónde están localizadas, lo que dificulta bastante el moverse fácilmente.

Ante esto vamos a enfocar este trabajo en intentar dar una solución con costes asequibles y método viables. Por lo que pensamos que es posible aunar tanto una parte software, que podría trabajar de manera autónoma, como una parte hardware.

¿Qué ofreceremos en forma de app móvil? Una aplicación que localice en un mapa las plazas de aparcamientos para personas con movilidad reducida de forma que puedan navegar fácilmente mediante el GPS de sus teléfonos hasta el lugar que más les interese. De esta forma podemos ayudar a que las personas puedan organizarse de manera más eficiente.

También, como segunda parte queremos desarrollar unos dispositivos hardware que, conectados a nuestra aplicación, detecten cuando una plaza pasa de estar libre a estar ocupada y viceversa, con lo que podamos enviar una señal a nuestra base de datos, actualizar el estado de la plaza y entonces poder mostrarlo en el mapa en tiempo real. Con esto podríamos optimizar más las rutas de personas discapacitadas que pueden ahorrarse el ir a un sitio si ya saben que estas plazas no están disponibles.

2. Estado de la cuestión

En este proyecto vamos a tocar varias cuestiones:

- Localización y puesta en conocimiento de plazas de aparcamiento para personas discapacitadas a todos los usuarios que lo requieran.
- Servir la información del estado de las plazas de aparcamiento (libres/ocupadas).
- Concienciación enfocada al grupo de personas que no respetan este tipo de plazas y las utilizan de forma indebida.

Antes de esta propuesta un usuario tiene a su alcance una serie de herramientas rudimentarias, las cuales en algunos casos son muy escasas e incluso, no hace muchos años (2013-2014), inexistentes, para la consulta de servicios para personas discapacitadas.

La evolución natural con el uso cotidiano de las TI sería tener herramientas en la palma de la mano ya sea con nuestro teléfono móvil, nuestro dispositivo portátil (ordenador, tablet...) e incluso un computador de sobremesa. Por lo que quizás veamos con extrañeza que, hasta hace pocos años en ciudades, de la Comunidad Valenciana, por ejemplo, no se poseyera, ni por parte de la Administración pública, un **listado** al que acudir para poder consultar la localización de las plazas de aparcamiento para personas con problemas de movilidad.

Con el tiempo estos problemas se han ido solucionando, pero no de forma centralizada. Esto significa que estos listados, bases de datos y/o almacenamiento de información y puesta a disposición de los usuarios se hace de forma desigual en todo el territorio español. Esto ocurre debido a que es un tema que administra cada ayuntamiento de forma autónoma, por lo que no se sigue una misma fórmula en todos los lugares donde existen estas plazas.

A pesar de esto, hoy en día, si buscamos rápidamente en internet podemos encontrar noticias en periódicos que hablan sobre normativas que se deberían seguir con el objetivo de localizar las plazas de aparcamiento [4]. Aun así, no encontramos gran cosa en la mayoría de los municipios y menos para toda España, como mucho en alguna web de ciudades grandes podemos descargar ciertos ficheros PDF o Excel con algunos datos.

Y aunque los anteriores avances no han sido suficientes para tener la información siempre al alcance (ya que para consultar estos medios puede ser un poco costoso o lento), sí que se han hecho avances y propuestas para solucionar las cuestiones antes mencionadas.

Haciendo un barrido rápido por Google [5] y las stores de las principales plataformas como pueden ser App Store [6] de iOS y el Play Store de Android, las aplicaciones más votadas o relevantes son, hoy, solamente una: Disable Park [7].

Sobre esta aplicación hemos recabado muchos datos gracias a que hemos podido atar cabos. Primero vimos qué empresa la había desarrollado y casualmente un amigo mío estuvo trabajando durante algún tiempo en esta. Este compañero consiguió ponerme en contacto con la persona encargada de que esta aplicación fuese creada y publicada: Ana Escudero.

Ana nos contó en una entrevista que ella es una chica con problemas de movilidad reducida y que su día a día está ligado a una silla de ruedas. Ella estudió en la Universidad de Alicante y junto con unos compañeros pensaron en una solución para la problemática de la localización de plazas para personas con movilidad reducida.

Ella nos contó que, por ejemplo, en el Ayuntamiento de Elche, hace unos años no poseían ningún registro de la localización de ninguna plaza. Para poder llevar a cabo adelante esta aplicación les puso en contacto con la persona encargada de pintar en el suelo las plazas.

Esta persona les dibujó en un plano de papel dónde se encontraban las plazas que él había pintado. Lo cual no significa ni que fuesen todas, ni que se acordarse de todas.

Por ciertos problemas con la empresa desarrolladora la aplicación no tuvo salida en iOS y la de Android por desgracia se encuentra sin mantenimiento. Primero porque las personas que idearon la aplicación no son desarrolladores y segundo porque el código de la aplicación no fue recuperado correctamente al acabar el desarrollo.

Ahora mismo el que alguna empresa o freelance coja el código que poseen y lo quieran trabajar para actualizar y resolver bugs es casi imposible por lo costoso en refactorización y también en coste monetario, ya que no poseen ingresos por la aplicación. Además, las ayudas que ingresaban por parte de organismos públicos se acabaron junto con el interés de los mismo.

La aplicación como veremos a continuación en un pantallazo es una aplicación que se utilizaba bastante, pero poco a poco ha ido teniendo malas puntuaciones por falta de mantenimiento.

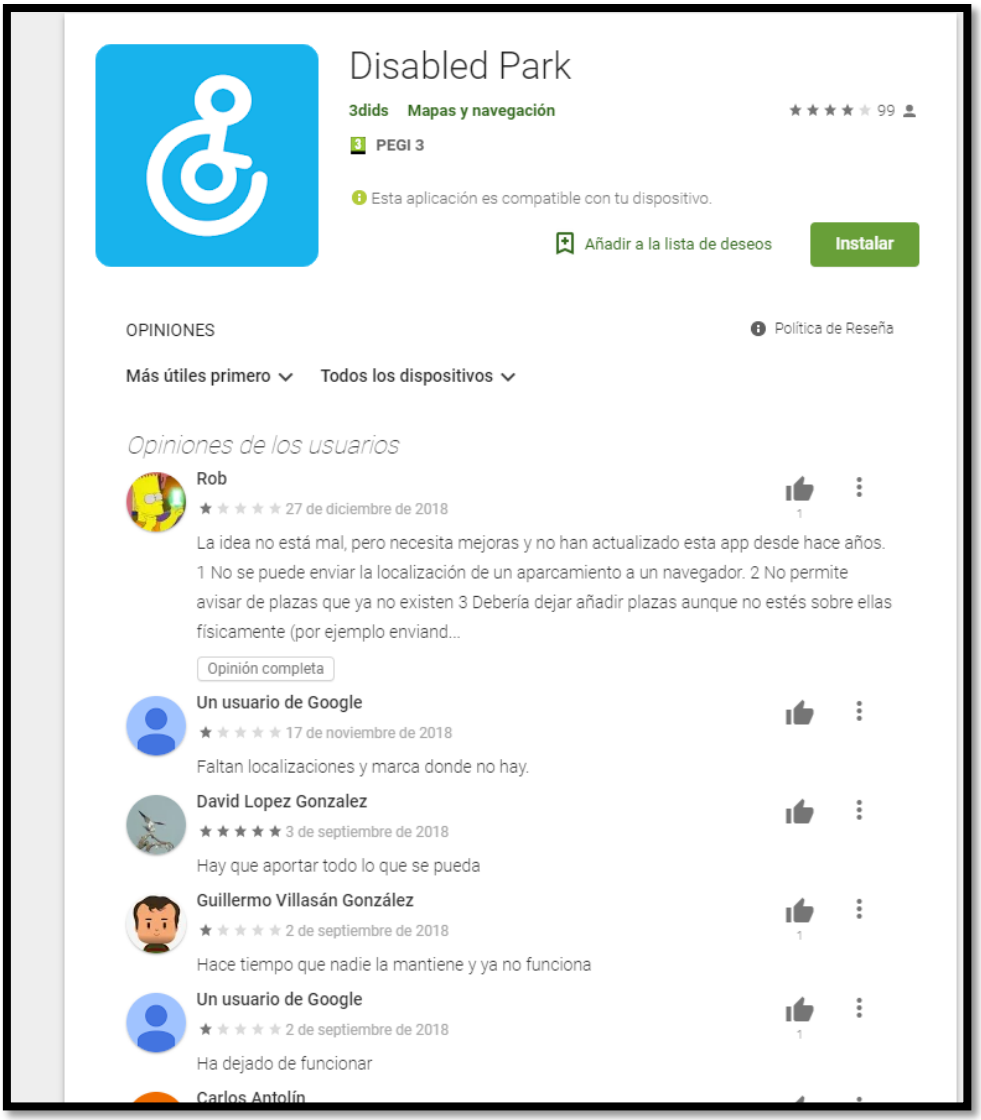

Figura 1 - Ejemplo aplicación (Google play)

Como podemos observar la aplicación tiene buena puntuación, pero últimamente no hace más que recibir malas puntuaciones, ya sea porque ha dejado de funcionar en algunos o todos los dispositivos o porque no tiene ciertas funcionalidades… Pero como podemos ver la comunidad es activa, por lo que es importante que podamos dar una solución a todas estas personas.

Finalmente hay que decir que los chicos de Disabled Park están encantados con nuestra propuesta, por lo que les pondremos al día al finalizar este proyecto.

Otras de las propuestas que encontramos en internet podría ser la plataforma, de habla inglesa, VMI imagine possibilities [8], donde se dedican a idear forma de facilitar a las personas su vida diaria: personalizando vehículos, dando soporte a las personas que lo necesitas…

En ciertos artículos de esta plataforma como, por ejemplo: *"Accessible Parking & Other Mobile Apps For Daily Travel Use":2017* [9], nos hablan sobre herramientas que podemos usar para orientarnos al buscar accesos para personas con problemas de movilidad. Ya que como bien explican el hecho de conocer ciertos sitios a los que nos movemos habitualmente no ayuda al hecho de que queramos movernos con nuestras personas a lugar por motivos de ocio, viajes, etc.

Por eso enumeran ciertas herramientas como:

- **BlueBadgeParking** [10]. Esta es una herramienta web internacional donde tenemos un mapa web. En este mapa tenemos muchos puntos donde se encuentran parkings para personas con movilidad reducida.

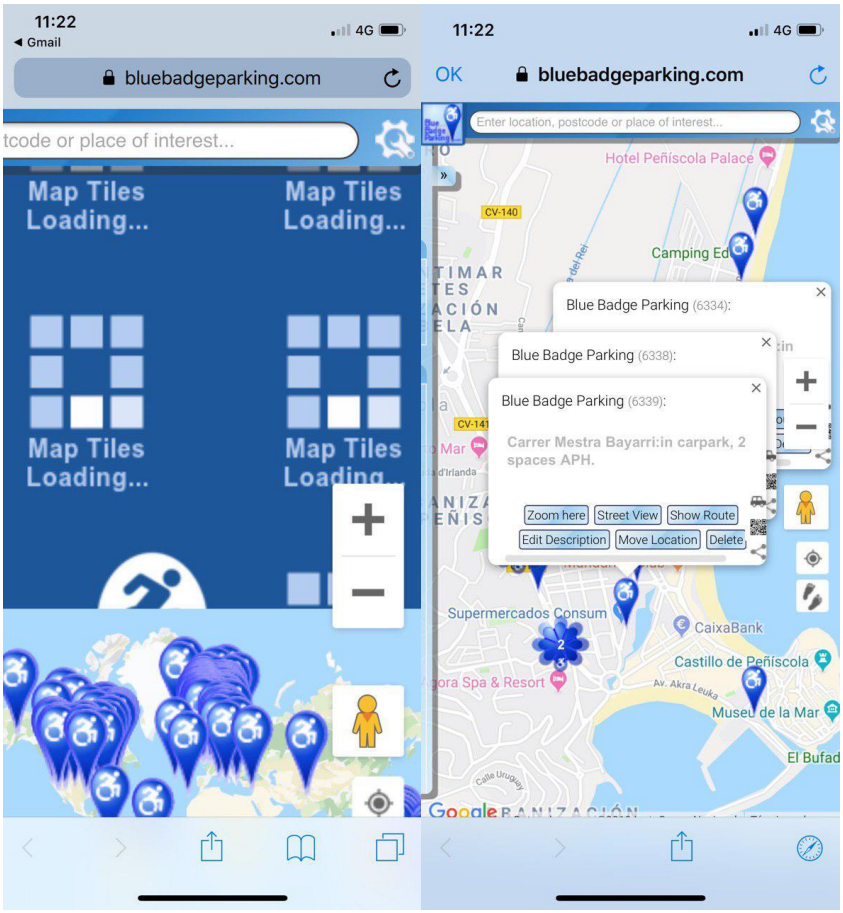

Figura 2 – BlueBadgeParking

¿Qué decir de esta herramienta? Podemos ver a primera vista que la usabilidad y la accesibilidad del producto no es lo principal.

Los botones son pequeños y están apiñados. Además, no queda claro para qué es cada uno. Los carteles de información se quedan abiertos. Igualmente, el mantenimiento para los móviles para no estar demasiado al día, pues he tenido muchos problemas para utilizar la aplicación con normalidad (pantallas bloqueadas, zoom cuando no lo pedíamos…).

Por lo que EnbleParking puede mejorar esta herramienta en ligereza, agilidad, usabilidad y claridad de interfaz.

- También nos hablan de las mejoras de Google Maps a la accesibilidad. Nos cuentan que en una actualización Google pone a disposición de los usuarios un sistema para añadir detalles de accesibilidad a distintos sitios por los que nos vayamos moviendo de forma personal.

  Aquí decir que para empezar no es algo intuitivo el llevar a cabo una contribución. Si intentamos llevar a cabo los pasos que nos indican en la web, a modo de guía, llegas a ciertos pasos en los que nos hemos sido capaces de acabar.

  Por lo que EnableParking se convierte en una herramienta móvil especializada donde tenerlo todo en unos sencillos pasos.

- **WheelMate** [11]. Otra herramienta que podemos descargar a nuestros celulares. La cual nos muestra en un mapa algunos puntos con plazas de aparcamiento y baños accesibles para usuarios con problemas de movilidad.

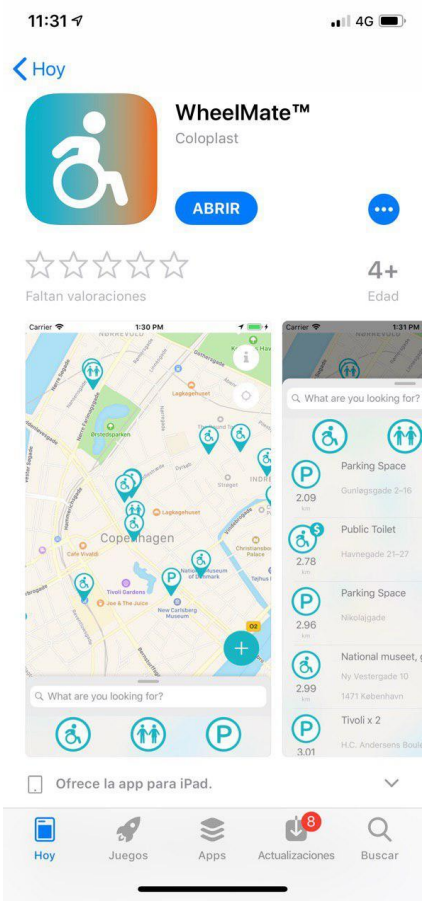

Figura 3 – WheelMate (App Store)

De esta herramienta hay que comentar que es una aplicación móvil internacional, pero que requiere que las personas, la comunidad se preocupe de nutrirla de lugares a tener en cuenta. Asimismo, en España no tiene un gran respaldo.

Con EnableParking podríamos nutrirnos de bases de datos que ya tenemos en otros proyectos hermanados por la causa, además de las asociaciones que están al corriente del desarrollo.

- Por último, **Parking Mobility** [12]. Una herramienta para denunciar el abuso de plazas de aparcamiento azules.

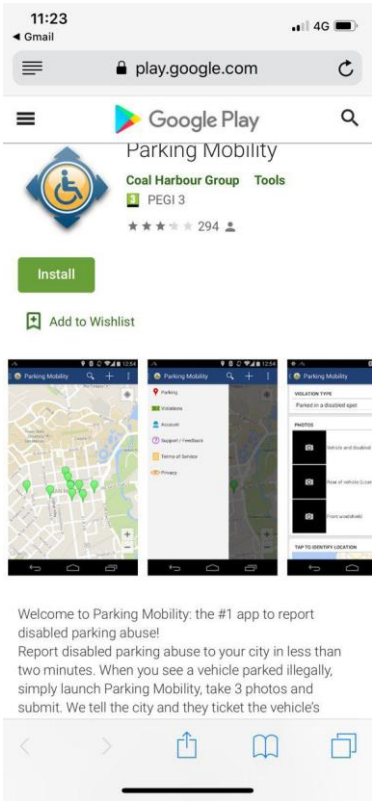

Figura 4 – Parking Mobility (App Store)

Esta última es una herramienta con origen en Canadá. Y está diseñada como, explican en la descripción, para denunciar los abusos a las plazas de aparcamiento azules mediante unas fotografías que se mandan por parte de los usuarios.

Esto no es uno de los objetivos principales de EnableParking, que lo que buscamos es ayudar a las personas a que su movilidad sea más ágil y cómoda. Donde pensamos que es mejor concienciar a la sociedad que estar denunciando continuamente. Aunque podría ser una posible futura mejora para nuestra aplicación, por lo que no lo descartamos.

Con todo lo comentado anteriormente podemos observar que a nivel local las soluciones para esta problemática son escasas y las que hay están dejando de tener soporte por motivos que se escapan un poco a las personas que idearon la solución. Pero esto lo podremos evitar con un análisis previo y una especificación marcando objetivos claros para saber a dónde queremos llegar y qué errores no queremos cometer.

Al mismo tiempo, al estar pensado como un producto opensource muchos de los problemas que tuvieron DisablePark podremos sortearlos.

Por otra parte, a nivel internacional vemos que, sí que se han llevado a cabo proyectos con buenos argumentos, pero que no ponen al día o actualizan la información en beneficio de los usuarios españoles. Y muchas de esas herramientas no contemplan los objetivos principales que marcamos con este proyecto.

Además, tenemos un valor de peso que queremos añadir, el hardware implantado en las plazas de aparcamiento. Esto da un valor añadido a esta aplicación, que de por si sola es muy útil, pero que la sustenta de mucha más utilidad.

3. Estudio de viabilidad

En este punto del documento vamos a ver un estudio de viabilidad del proyecto con el objetivo de analizar la viabilidad, los puntos fuertes y los puntos que pueden convertirse en riesgos. Para esto haremos uso de *Lean Canvas* [13], una herramienta de visualización de modelos de negocio "ligero", muy usado en entornos startup e ideas innovadoras.

a. Lean Canvas

Lean Canvas es una herramienta perfecta para el análisis de productos que están siendo creados y poseen un carácter innovador, ya sea porque no existe una solución parecida en el mercado o porque es un producto que ofrece alguna mejora con respecto a algo ya publicado.

El objetivo de esta herramienta es poder analizar la solución o el proyecto desde diversos puntos de vista interesantes como veremos en la siguiente figura:

Figura 5 - Lienzo Lean Canvas (Innokabi)

En este apartado tenemos que dejar claro que EnableParking es un producto enfocado a ayudar a las personas que lo necesite y que, por lo tanto, no se busca un beneficio económico, por lo que hay que ver el éxito en el hecho de que la comunidad, los ayuntamientos y asociaciones se hagan eco del sistema y trabajen en él de forma que lo puedan adaptar a sus necesidades.

Dicho lo anterior vamos a seguir este lienzo y vamos a ir viendo cómo sería el enfoque desde un proyecto de esta índole, de forma que podamos conocer un poco mejor el proyecto. Y el orden de los puntos no va a seguir ningún patrón establecido. Si no que trataremos un orden que pensamos que se explica mejor.

Segmento de clientes

Las claves del éxito de EnableParking es el poder consultar en cualquier momento el estado de las plazas de aparcamiento, saber dónde están y saber su estado (si tienen instalado el dispositivo). Por lo tanto, los clientes potenciales serían las personas con problemas de movilidad y también las personas, familiares y amigos, que les ayudan en su día a día a moverse con un vehículo.

Los early adopters son los primeros usuarios reales que usarán la aplicación y que nos darán un feedback sobre las funcionalidades y acciones realizadas con este sistema. Por tanto, estableceremos como early adopters a los integrantes de la asociación AMFI, los cuales son los primeros interesados en ello y también podrían ser los chicos y chicas de DisablePark. Además, ellos son unos aliados potentes para nuestro sistema, puesto que han utilizado una aplicación similar, pero con las restricciones de no poder implementar hardware en las propias plazas.

Problema

Los problemas de nuestros clientes son varios.

- Por un lado, no saber todos los sitios donde pueden disponer de una plaza de aparcamiento apta para sus necesidades.
- Al mismo tiempo, para ellos, moverse y aparcar en ciertos sitios requiere un tiempo y esfuerzo superior a la media, por lo que ayudarles a saber si hay plazas de aparcamiento libres en el sitio al que se dirigen es muy útil ya que pueden reaccionar y saber dónde ir en cada caso.

Las soluciones que proponemos a estos problemas son bastante innovadoras, ya que tendríamos un sistema de localización de plazas de aparcamiento con el valor añadido de dispositivos que nos den el estado (libre/ocupado) en tiempo real y que se pueda consultar desde cualquier sitio, además de que nos dirija mediante una navegación hasta ellas.

Riesgos

El mayor de los riesgos que tendríamos en EnableParking es que a la larga las personas no utilicen una aplicación que se ha hecho para ser implantada con el objetivo de ayudar y hacer la vida de las personas más sencilla.

Como también que las asociaciones y las administraciones públicas no inviertan algo de tiempo y esfuerzo en que sea más y más conocida. Ya que ellos son los medios de difusión más importantes puesto que son los que más en contacto están con las personas que van a dar buen uso de este sistema.

Otro de los riesgos es que los dispositivos hardware no consigan el objetivo. Bien porque sean demasiado caros de implementar e instalar correctamente y por lo tanto los

inversores no quieran hacerlo, bien porque la sociedad no se conciencie en el correcto funcionamiento de estos y nos los traten con cuidado.

Por último, el que la base de datos que hemos elegido no quiera ser mantenida por temas de costes que, aunque no muy elevados, puede no interesar.

Proposición de valor única

Único sistema que facilitará la utilización de plazas de aparcamiento para personas con movilidad reducida de forma que puedan ser consultadas (estado de la plaza), localizadas y ayude a dirigirse a ellas de forma eficiente.

Canales

Los primeros medios de difusión que nos viene a la cabeza podrían ser asociaciones como AMFI, ya que son las personas que están al pie del cañón con las personas que tienen problemas de movilidad. Además, ellos pueden hacer uso de sus redes sociales, las cuales siguen personas que son el 'target' de esta aplicación.

Estructura de costes

Los principales costes serán el coste del desarrollo de la aplicación y el mantenimiento posterior de la misma.

Asimismo, utilizaremos como base de datos un servicio de Google, que, aunque al principio le damos un uso gratuito, a la larga puede requerir de cierto pago.

Lo que podemos decir es que el volumen puede que no sea excesivo, por lo que el coste del servicio debería ser aceptable.

4. Planificación

Una buena planificación siempre es necesaria y sobre todo cuando queremos sacar un proyecto adelante en un tiempo determinado.

Esta etapa de estudios ha sido llevada a cabo al compaginada con mi trabajo a jornada completa por lo que tuve que ser sincero conmigo mismo y opté por una planificación tomando conceptos de una metodología ágil como puede ser SCRUM.

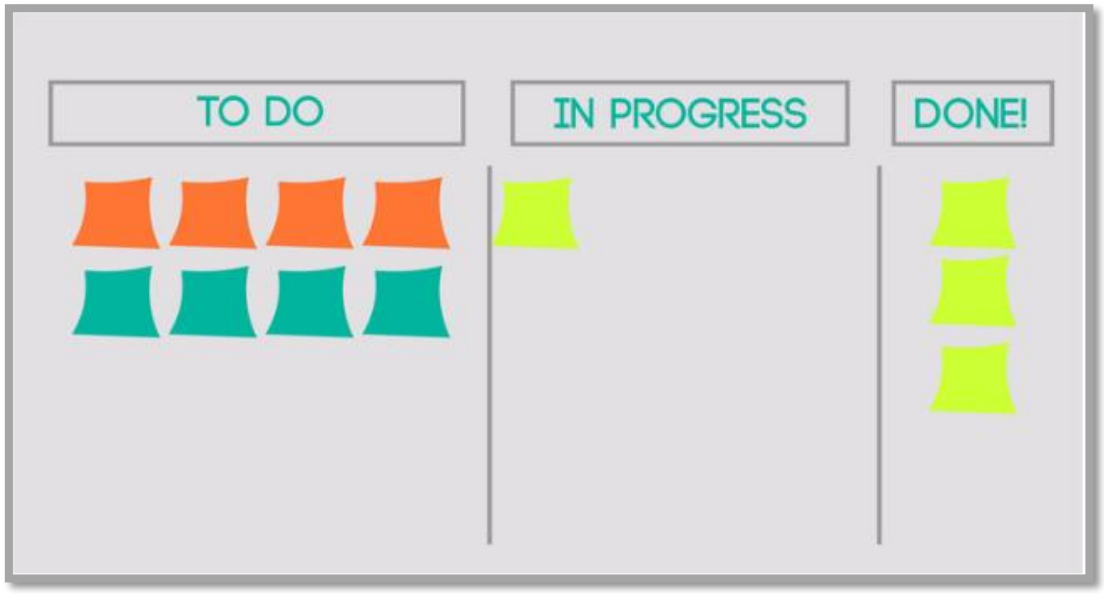

Figura 6 – Ejemplo de tablero SCRUM (Google imágenes)

No tomé plazos de tiempo tipo sprint pues sabía que el trabajo no me iba a dejar organizarme jornadas de 8 horas ni mucho menos, pero sí que rellené un tablero scrum. Esto es un tablero con tres columnas (To Do, In progress, Done) las cuales rellenaba con pequeñas tareas que pudiera llevar a cabo en un par de tardes o una semana, de forma que pudiera ir avanzando poco a poco y conseguir evitar la desmotivación por el hecho de tener pocas horas a la semana para poder dedicar al proyecto. Además, era mi agenda, pues tenía todas mis ideas que se iban quedando como tareas en la columna "To Do".

Cada tarea está etiquetada de una forma concreta:

- Verde → desarrollo / codificación
- Azul → diseño
- Lila → Memoria / redacción
- Amarillo → Leer documentación
- Rojo → la tarea implica dos tardes (4 a 6 horas), 4 tardes máximo
- Negro → desvío de horas

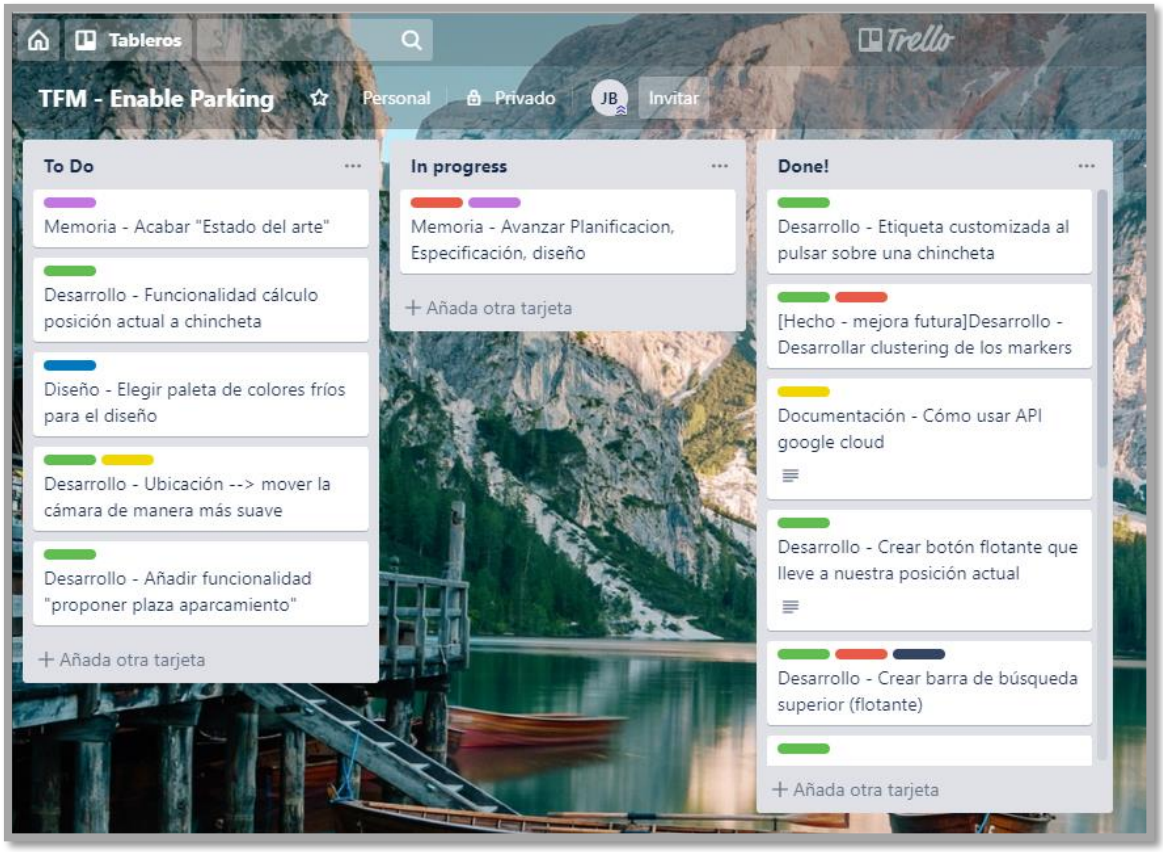

Figura 7 – Tablero SCRUM (Fuente propia)

Gracias a la herramienta de planificación Trello [14], de un vistazo, podría saber a qué me iba a dedicar en la semana actual. Una vez acaben las tareas de la columna “To Do” se revisa el proyecto y las tareas completadas y se rellena de nuevo la primera columna hasta acabar con los objetivos principales establecidos por el análisis e ideas obtenidas para detalles a lo largo del desarrollo.

5. Objetivos

En este punto veremos los objetivos marcados, es decir el punto o la meta que queremos alcanzar dentro del proyecto habiendo visto lo anterior en este documento. A la hora de proponernos unos objetivos debemos tener en cuenta las pegas y dificultades que nos podemos encontrar por el camino, de forma que decidamos bien hasta dónde podemos llegar en un tiempo dado, además de los problemas que nos iremos encontrando durante el desarrollo del proyecto.

Nuestro objetivo principal es crear una aplicación que permita a los usuarios saber dónde pueden disponer de una plaza de aparcamiento para personas con movilidad reducida allá donde quieran buscar. Para ello nos basaremos en todo lo visto a lo largo del documento y nos apoyaremos en lo existente en el mercado, de forma que podamos darle una vuelta de tuerca y que los dispositivos hardware implementados nos den la información de si una plaza está ocupada o no.

Estos objetivos dependen de dos trabajos, una parte hardware y otra parte software. Esta última es en la que nos centramos ya que es el objetivo principal de este máster, aunque un objetivo importante para nosotros es mejorar lo que otros no pudieron llegar a hacer por culpa de un diseño previo no enfocado al hardware posterior. Por lo que prepararemos el sistema para que sea acoplable a la parte hardware de forma sencilla, además de usar una tecnología de moda y con mucha documentación de forma que cualquiera que llegue después pueda adaptarse al proyecto, código… de forma simple y así conseguir que el proyecto no caiga en saco roto.

Este objetivo será satisfecho si llegamos a una aplicación móvil que sea funcional con datos reales de las plazas existentes y además nos aporte información en tiempo real de su estado. Además de conseguirlo sería una buena idea el poder, por un lado, guiar a las personas hasta su objetivo con una interfaz amigable y sencilla, por otro lado, conseguir que los usuarios puedan interactuar con el sistema de forma que pudiera convertirse en una herramienta con tintes colaborativos.

En caso de conseguirlo sería una herramienta perfecta para que en las asociaciones de personas con movilidad reducida y en los ayuntamientos se diera a conocer de forma que pasase a formar parte de la vida cotidiana de muchísimas personas.

En cuanto a las fechas marcadas para la finalización del proyecto está marcado por la evaluación de este en septiembre de 2019. Pero hay que tener en cuenta que lo que queremos conseguir al menos al finalizar este tiempo, en el peor de los casos, es un prototipo preparado para que la parte hardware sea implementable por cualquiera y la aplicación móvil sea usable y útil, aunque tenga menos extras.

Por lo tanto, lo que determinará si hemos conseguido llegar a buen puerto al final de la etapa es:

- Aplicación usable, en un principio, en dispositivos Android.
- Un mapa navegable en el que mostremos todas las plazas almacenadas en nuestra base de datos en la nube.

- Dotar a la aplicación de la automatización del cambio de estados de las plazas de aparcamiento reflejadas en el mapa mostrado.

Tras esto el proyecto quedará de forma que podamos seguir trabajando en él, mejorándolo poco a poco mediante actualizaciones, creando funcionalidades útiles y controladas, además gestionar la BD para tener al día las plazas de aparcamiento existentes (consiguiendo el apoyo de la comunidad mediante alguna funcionalidad) y conseguir llegar a todas las plataformas.

6. Análisis y especificación

En este apartado nos centraremos en la problemática concretamente. De forma que vamos a definir y analizar unos requisitos que, pensamos, van a ayudar a solventar la problemática. Para llevar a cabo esta tarea es muy útil utilizar una herramienta estandarizada de forma que dejemos los mínimos elemento o ninguno sin definir. Y para ello usaremos el estándar IEEE 830 [15].

Este nos guiará a formular especificaciones software, de forma que enumeremos lo que el sistema a desarrollar hará y al finalizar tendremos una lista de los requerimientos funcionales y no funcionales.

a. Alcance

Diseño, desarrollo e implantación del sistema EnableParking.

Será una aplicación web que permitirá a los usuarios consultar dónde y en qué estado están las plazas públicas para personas con problemas de movilidad reducida, también denominadas personas con discapacidad. Además, los usuarios podrán:

- Crear una ruta desde su posición hasta la plaza señalada.
- Enviar la posición en la que están por tal de compartir donde existe una plaza que no contempla la app.
- Marcar una plaza como “demandada” para que los demás usuarios vean que alguien está interesado en llegar en poco tiempo para darle uso.

Con este proyecto se pretende facilitar la vida a aquellas personas que hacen uso de este tipo de plazas, que gracias a este tipo de propuestas pueden conocer dónde existen y pueden hacer uso de ellas, puesto que lo normal es que no exista un censo de ellas creado por los ayuntamientos.

b. Definiciones, acrónimos y abreviaturas

Nombre	Descripción
BD	Base de datos
ERS	Especificación de Requisito software
RF	Requisito funcional
RNF	Requisito No funcional

Tabla 1 - Abreviaturas

c. Referencias

Referencia	Título	Fecha
Standard IEEE - 830 - 1998	Especificación de Requisitos según el estándar de IEEE 830	22 octubre de 2008

Tabla 2 - Referencias

d. Perspectiva del producto

La aplicación EnableParking está diseñada para trabajar en entornos android por lo prolífico del sistema operativo, pero en un futuro habría que pensar en desarrollar una aplicación en Swift de forma que abarcase también a los usuarios de un dispositivo iOS.

Además, busca ser una aplicación con una interfaz limpia y sencilla de forma que cualquier persona, desde los más experimentados a las personas que casi no utilizan el teléfono móvil puedan utilizarla sin problemas.

e. Evolución previsible del sistema

En un inicio la aplicación está diseñada para ayudar a las personas a conocer y llegar a plazas de aparcamiento que necesiten para sus necesidades, pero no se descarta que en un futuro la aplicación pueda ser más social, de forma que se pueda:

- Dar opiniones sobre las plazas: si el lugar es cómodo, si es espacioso, si los accesos de alrededor son buenos… dar una puntuación.
- Conseguir que la aplicación nos avise al entrar de que están ocupando una plaza que nos pertenece por matrícula.
- O notificar si alguna de nuestras plazas favoritas está libre u ocupada.

Todo esto dependiendo del impacto de la implantación tanto de la aplicación móvil por medio de algún tipo de publicidad por parte de los ayuntamientos haciendo llegar esta información a las personas interesadas.

f. Requisitos específicos

Interfaces externas

Los colores que se utilizarán para el diseño de la interfaz de usuario serán:

- Colores limpios sin gradientes
- Tonos Material Design
- Paleta de azules

La interfaz estará en vertical

Deberá adaptarse a las dimensiones del dispositivo Android utilizado

Requisitos funcionales

Id. del requisito:	RF 01
Nombre del requisito:	Petición permisos para obtener ubicación
Características:	La aplicación usará los datos de ubicación del dispositivo para marcarla en el mapa.
Descripción del requisito:	El sistema pedirá permisos al usuario para tomar los datos de ubicación del dispositivo mediante un popup de confirmación “aceptar” o “cancelar” y almacenará la respuesta.
Prioridad del requisito:	Alta

Id. del requisito:	RF 02
Nombre del requisito:	Mapa interactivo
Características:	Mapa con las marcas de las plazas de aparcamiento.
Descripción del requisito:	La aplicación mostrará el mapa de la zona con el que se podrá interactuar utilizando algún API, de forma que puedan interactuar con él y sus marcadores de plazas de aparcamiento.
Prioridad del requisito:	Alta

Id. del requisito:	RF 03
Nombre del requisito:	Ubicación usuario
Características:	Señalar la ubicación del dispositivo.
Descripción del requisito:	El mapa marcará la ubicación del dispositivo del usuario en tiempo real, de forma que pueda saber dónde se encuentra.
Prioridad del requisito:	Alta

Id. del requisito:	RF 04
Nombre del requisito:	Barra de búsqueda
Características:	El usuario dispondrá de una barra de búsqueda.
Descripción del requisito:	El sistema mostrará una barra de búsqueda en la parte superior del mapa mediante el que podrá hacer búsquedas de lugares. Será diferenciable por algún icono o hint y al pulsar desplegará un teclado con el que introducir las palabras.
Prioridad del requisito:	Alta

Id. del requisito:	RF 05
Nombre del requisito:	Menú inferior
Características:	Una barra con varios botones.
Descripción del requisito:	El sistema tendrá una barra inferior con la que podrá interactuar con las opciones. Los botones serán entre 3 y 4, cuadrados y cada uno de ellos llevará a cabo una acción.
Prioridad del requisito:	Media

Id. del requisito:	RF 06
Nombre del requisito:	Botón flotante para tipo de mapa
Características:	El usuario podrá cambiar la representación del mapa.
Descripción del requisito:	La aplicación tendrá un botón flotante mediante el cual el usuario podrá seleccionar el tipo de mapa a mostrar (normal/ satélite).
Prioridad del requisito:	Baja

Id. del requisito:	RF 07
Nombre del requisito:	Consultar plaza de aparcamiento
Características:	El usuario podrá ver qué plazas de aparcamiento existen.
Descripción del requisito:	El sistema representará mediante marcadores las plazas de aparcamiento registradas en la BD.
Prioridad del requisito:	Alta

Id. del requisito:	RF 08
Nombre del requisito:	Ruta hasta plaza aparcamiento
Características:	El usuario podrá trazar una ruta hasta la plaza de aparcamiento.
Descripción del requisito:	El sistema permitirá mediante la propia aplicación o mediante Google maps trazar una ruta hasta la plaza seleccionada. De forma que podamos seguir la línea hasta el punto indicado
Prioridad del requisito:	Media

Id. del requisito:	RF 09
Nombre del requisito:	Dirección plaza aparcamiento
Características:	El usuario podrá saber en qué calle se encuentra la plaza.
Descripción del requisito:	El sistema permitirá mediante una etiqueta saber dónde se encuentra la plaza pulsada por el usuario.
Prioridad del requisito:	Media

Id. del requisito:	RF 10
Nombre del requisito:	Color marcadores plazas
Características:	El usuario sabrá si la plaza está ocupada o libre.
Descripción del requisito:	La aplicación cambiará de color el marcador cuando la plaza quede libre o está ocupada.
Prioridad del requisito:	Alta

Id. del requisito:	RF 11
Nombre del requisito:	Clustering de marcadores
Características:	La aplicación agrupará marcadores.
Descripción del requisito:	La aplicación podrá ir agrupando y desagrupando marcadores a medida que alejamos o acercamos el zoom del mapa.
Prioridad del requisito:	Baja

Id. del requisito:	RF 12
Nombre del requisito:	Detalle Parking
Características:	Pantalla de detalle de la plaza
Descripción del requisito:	El usuario podrá consultar una pantalla de detalle de la plaza que le interese
Prioridad del requisito:	Baja

Id. del requisito:	RF 13
Nombre del requisito:	Puntuación plaza
Características:	Votos positivos/negativos del parking
Descripción del requisito:	El usuario podrá consultar en el detalle la puntuación dada por los usuarios a la plaza que busque
Prioridad del requisito:	Baja

Id. del requisito:	RF 14
Nombre del requisito:	Puntuación plaza
Características:	Votos positivos/negativos del parking
Descripción del requisito:	El usuario podrá consultar en el detalle la puntuación dada por los usuarios a la plaza que busque
Prioridad del requisito:	Baja

Id. del requisito:	RF 15
Nombre del requisito:	Acerca de
Características:	Información acerca de la aplicación
Descripción del requisito:	El usuario podrá acceder a una pantalla de información acerca de la aplicación
Prioridad del requisito:	Baja

Requisitos no funcionales

Id. del requisito:	RNF 01 / BD 01
Nombre del requisito:	Base de datos
Características:	La aplicación tendrá una base de datos.
Descripción del requisito:	La app contará con una base de datos NoSQL siguiendo el esquema que seguirá en el apartado de diseño.
Prioridad del requisito:	Alta

Id. del requisito:	RNF 02 / BD02
Nombre del requisito:	Tiempo real
Características:	Actualización en tiempo real.
Descripción del requisito:	La aplicación escuchará cambios en tiempo real del servidor cuando esté en primer plano, de forma que esté siempre actualizado el estado de las plazas.
Prioridad del requisito:	Alta

Id. del requisito:	RNF 03
Nombre del requisito:	Sin conexión
Características:	Trabajar sin conexión.
Descripción del requisito:	Firestore ayudará a que la aplicación pueda trabajar sin conexión hasta que se restablezca.
Prioridad del requisito:	Baja

7. Diseño del sistema

Como hemos podido ir viendo a lo largo del documento EnableParking se pensó como una herramienta online para que funcionase en nuestros dispositivos móviles. Dicho esto, se propuso hacer un primer prototipo, y en la parte software se decidió una tecnología como Android de forma que pudiéramos abarcar la mayor parte del mercado de los teléfonos móviles y Firebase Firestore para la parte del almacenamiento de datos dado que permite a todos los clientes estar actualizados en todo momento.

Por otro lado, y con el apoyo de nuestros compañeros de Artefactos (UA) se ha decidido usar tecnologías Hardware como ESP32 [16] como dispositivos controladores de las plazas de aparcamiento mediante los cuales enviar los datos a la base de datos para la actualización del estado de la propia plaza.

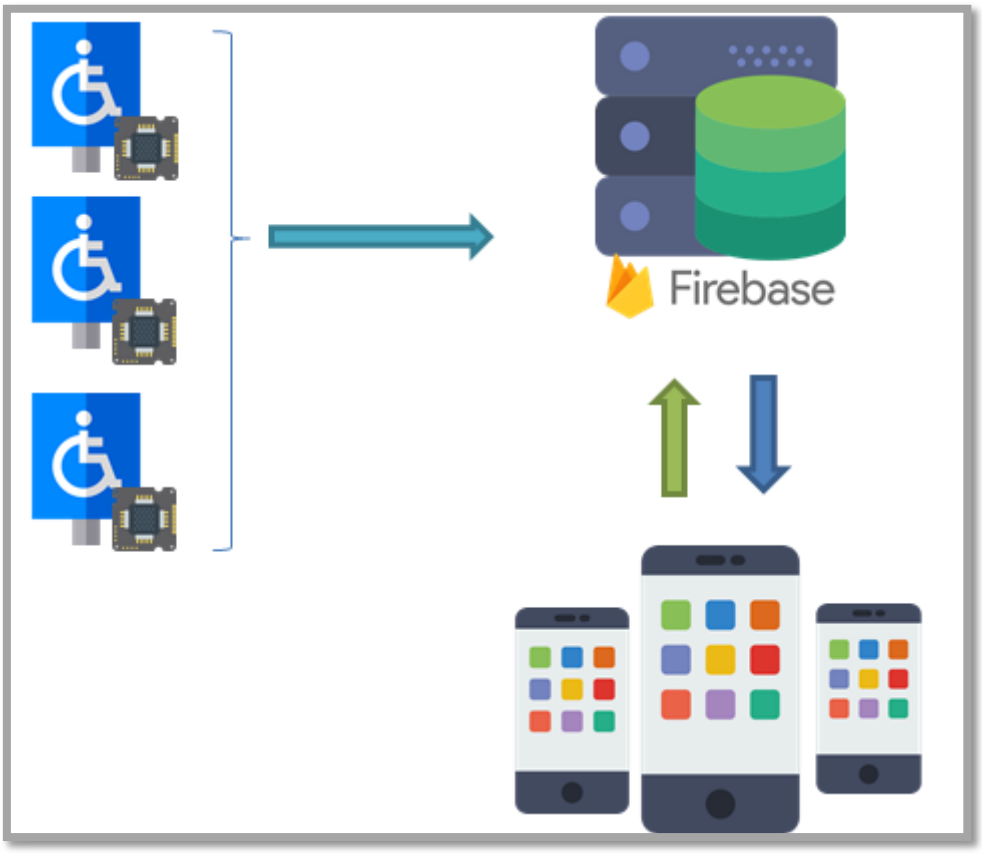

Figura 8 – Esquema arquitectura EnableParking (Fuente propia)

a. Diseño de persistencia

En este apartado vamos a hablar de la persistencia y para ello vamos a ver el tipo de base de datos utilizada y todo lo que tiene que ver con el modelo de datos utilizado.

En cuanto al modelo de datos y el por qué se ha elegido una solución y no otra, una de las razones por la que una base de datos no relacional (NoSQL) puede ser muy práctica en esta aplicación es que las entidades como puedan ser las plazas de aparcamiento no van a cambiar en exceso y el volumen de datos no va a ser muy grande. Y por ello esta solución facilita o simplifica el trabajo.

Por otro lado, en las NoSQL no tenemos entidades sobre las que hacer una transacción atómica [17]; lo que ocurrirá es que si intentamos hacer un cambio en 3 entidades distintas vamos a tener que realizar 3 llamadas diferentes a la BD y esto puede traer problemas cuando más de un cliente hacen una solicitud sobre el mismo dato. Pero en nuestra aplicación, en principio, no debería darse un escenario de este tipo, pues la BD será administrada por un responsable, de forma que será supervisada a la hora de hacer alguna modificación.

El pro es que la atomicidad es cara a nivel de máquina, pues hay que tener varios aspectos en cuenta a la hora de manejarla en cada transacción, en cambio ganamos en velocidad además de estar enfocado a los dispositivos móviles (**mobile-first).**

Dejando esto de lado, otra razón de fuerza para elegir una tecnología como Firebase Firestore es que tiene una implementación sencilla y dirigida a las nuevas tecnologías. Es relativamente sencillo implementar estas librerías y las conexiones desde Android Studio y conseguimos con esto que la base de datos, por lo tanto, el almacenamiento, esté en la nube de forma que podremos acceder a ello desde cualquier dispositivo y de manera rápida gracias a los servicios que nos brinda Google, con lo que conseguimos una buena persistencia.

Además, disponemos de la característica del Realtime database, que como el propio Google define es una base de datos donde los datos se almacenan con un formato (no tiene por qué ser una forma única) y se sincronizan en tiempo real con cada cliente conectado. Y cuando compilamos aplicaciones multiplataformas (iOS, Android, Web…) todos comparten una instancia de Realtime Database y reciben actualizaciones automáticamente con los datos más recientes.

Sin dejar de lado una fachada que hemos codificado en la parte móvil para que la incorporación de datos desde un modelo por documentos como es el Firestore se nos automatice a la hora de trabajar con objetos en la parte JAVA.

Funciones clave:

- **Tiempo real**: En lugar de solicitudes HTTP típicas, Firebase Realtime Database usa la sincronización de datos (cada vez que cambian los datos, los dispositivos conectados reciben esa actualización en milisegundos). Proporciona experiencias colaborativas y envolventes sin pensar en el código de red.

- **Sin conexión**: Cloud Firestore almacena en caché datos que usa tu app de forma activa, por lo que la app puede escribir, leer, escuchar y consultar datos, aunque el dispositivo se encuentre sin conexión. Cuando el dispositivo vuelve a estar en línea, Cloud Firestore sincroniza todos los cambios locales de vuelta a Cloud Firestore
- **Acceso desde dispositivos cliente:** Se puede acceder a Firebase Realtime Database directamente desde un dispositivo móvil o un navegador web; no se necesita un servidor de aplicaciones. La seguridad y la validación de datos están disponibles a través de las reglas de seguridad de Firebase Realtime Database: reglas basadas en expresiones que se ejecutan cuando se leen o se escriben datos.
- **Escalamiento en varias bases de datos:** Con Firebase Realtime Database y sus planes, puedes satisfacer las necesidades de datos de la app a gran escala: podrás dividir la información en diversas instancias de bases de datos dentro del mismo proyecto de Firebase. Usa Firebase Authentication para el proceso de autenticación. Controla el acceso a la información de cada base de datos. Para ello, se pueden usar las reglas personalizadas de Firebase Realtime Database en cada una de las instancias de la base de datos.
- **Consultas expresivas:** podemos usar consultas para recuperar documentos específicos o para recuperar todos los documentos de una colección que coinciden con los parámetros de la consulta. Las consultas pueden incluir varios filtros en cadena y combinar los filtros con criterios de orden. También se indexan, por lo que el rendimiento de las consultas es proporcional al tamaño de tu conjunto de resultados, no del conjunto de datos.

Como hemos visto Cloud FireStore es una BD almacenada en la nube a la que podemos acceder desde diversos tipos de dispositivos, pero ¿cómo funciona?

Partimos de un modelo NoSQL donde los datos se almacenan en documentos que contienen campos con valores. Estos campos pueden ser de diversos tipos: booleanos, números, etc.

Figura 9 – Documento plaza aparcamiento (fuente propia, Firebase)

Estos documentos se almacenan a su vez en colecciones que son como carpetas que contienen los documentos que forman parte de esta y les da una característica por la que diferenciarlos, lo cual ayuda a organizarlos pues que se pueden anidar:

ej.: colección → documento → colección → documentos

De forma que los documentos admiten subcolecciones como si fuesen un tipo complejo de datos y de esta forma referenciarlos. De forma que convierte este tipo de base de datos en una de fácil escalado a medida que vaya creciendo la aplicación.

En nuestra base de datos tenemos una colección que denominamos plazas y dentro tenemos muchos documentos, uno por plaza de aparcamiento registrada.

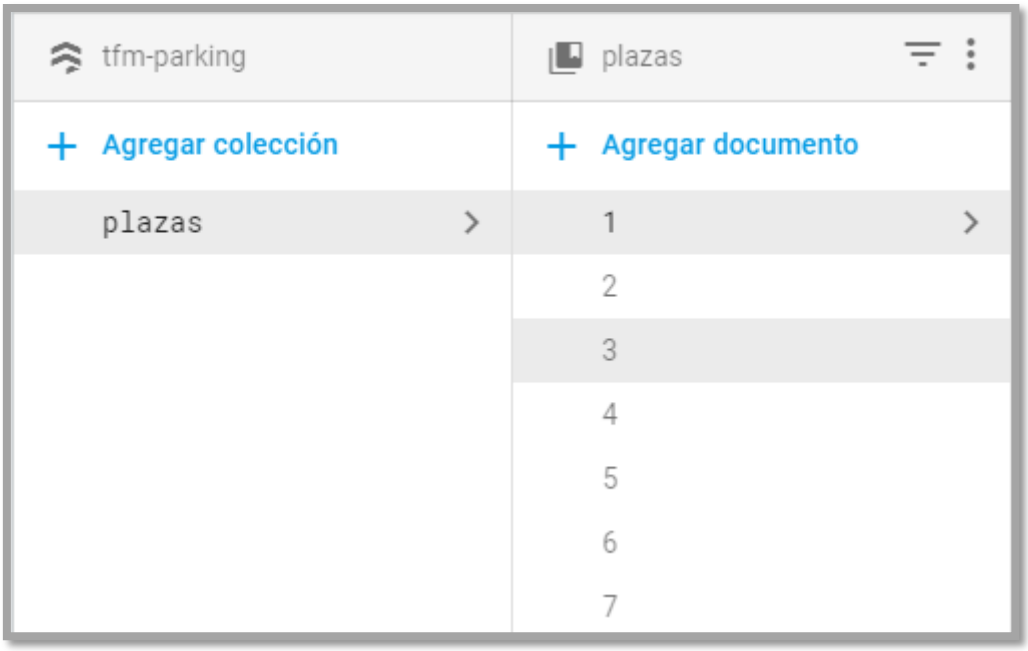

Figura 10 – Colección plazas aparcamiento (Fuente propia, Firebase)

Como hemos dicho las plazas de aparcamiento están registradas cada una en un documento, dentro de estos documentos tenemos todos los datos que la definen, las cuales siempre podemos modificar si quisiéramos escalar añadiendo o incluso quitando datos.

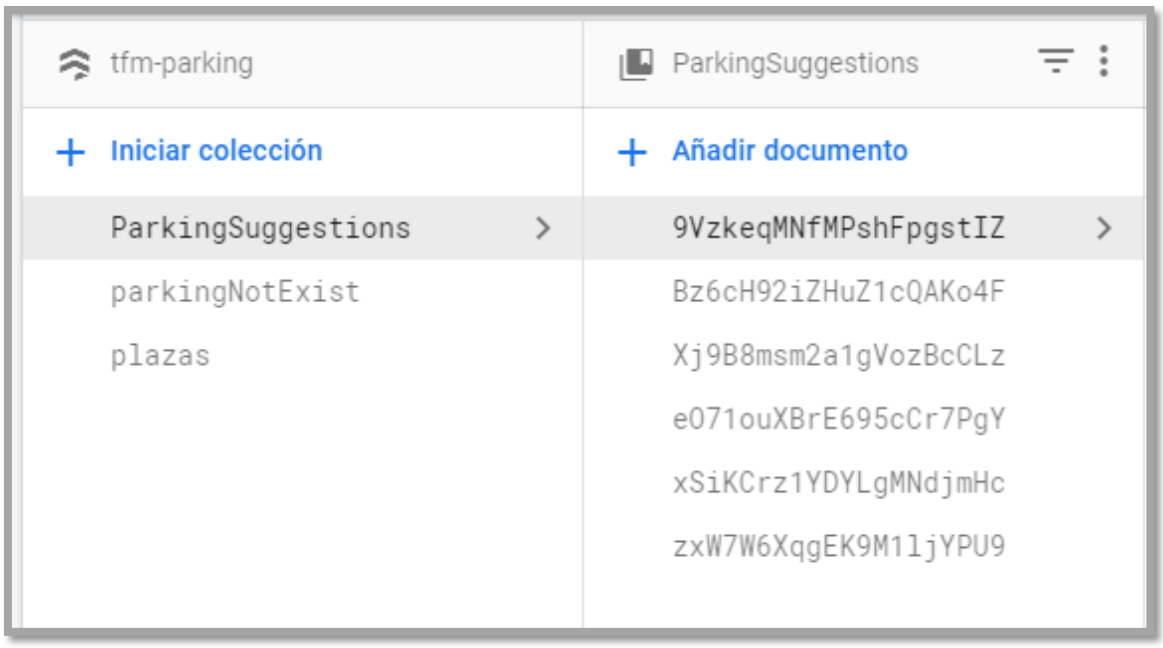

Figura 11 – Colección sugerencias de plazas de aparcamiento

Esta colección forma parte de la funcionalidad para que los usuarios envíen las sugerencias sobre nuevas plazas de aparcamiento o sobre plazas que no tengamos registradas en nuestro sistema, ya que almacenamos todas las coordenadas para un posterior tratamiento de los datos.

Figura 12 – Ejemplo de documento: sugerencia de plaza

b. Diseño de interfaces

El diseño de interfaces es el apartado donde plasmamos todos los requisitos que se plantean en el punto sobre especificación, es decir es en esta parte donde diseñamos soluciones mediante las cuales las personas podrán usar nuestra aplicación.

Al final de esta parte de la memoria deberíamos haber visto los diseños de cómo quedaría la aplicación al final de este desarrollo, de forma que se respeten en mayor medida, con lo que al cumplirlos también cumpliríamos los requisitos planeados.

En este apartado vamos a ver también un tema muy importante, tanto que en carreras como la ingeniería multimedia se tienen asignaturas que lo abarcan de forma profunda, esto es la usabilidad y la experiencia de usuario [18]. Estos conceptos poseen unos principios a la hora de analizar una aplicación que nos ayudan a conseguir unas mejores aplicaciones y que en mi opinión debería ser una materia obligatoria en todas las carreras que tengan que ver con el desarrollo del software. Además de que si llevamos a cabo un diseño de las pantallas finales conseguiremos agilizar a la hora del desarrollo.

En este proyecto el diseño era el punto flojo ya que lo he trabajado muy poco anteriormente, por eso he pedido ayuda a una compañera de multimedia y así acercarme un poco a los principios antes nombrados y que veremos algunos que me parecen destacables.

La usabilidad en resumen se considera una medida de calidad del producto y nos habla sobre cómo el producto, aplicación móvil en este caso, debe centrarse en que el usuario preste su atención en realizar la tarea para la que ha sido diseñada la app mucho antes que en la misma. Por lo que se busca medidas para que la aplicación sea fácil de utilizar y el usuario no tenga que pensar demasiado cómo usarla, por lo que entre otras características deberíamos buscar:

- Poco coste de aprendizaje
- Poca asistencia necesaria para el usuario
- El usuario se equivoca poco
- Mejora la calidad de vida de los usuarios

Por otro lado, en cuanto a la accesibilidad vamos a centrarnos en la parte que se centra en hacer que el diseño de la aplicación permite a las personas: percibir, entender, navegar e interactuar sin problemas.

Por ello y para no ahondar más en estos conceptos hemos optado por una aplicación con un diseño simple de utilizar de forma que a los usuarios no les cueste entender lo que están viendo y utilizando. Además, una pantalla principal con pocos botones y un mapa que abarca la mayor parte de la pantalla.

Por otro lado, la interfaz es muy parecida a las que solemos ver en aplicación de uso cotidiano como puede ser Google Maps [19] o Maps/Mapas [20], con botoneras sencillas como podemos ver en cualquier red social, como dictan los estándares del diseño de aplicaciones, como puede ser el Material design.

Siguiendo con esto el esquema de colores que hemos creado (punto 7.3 Guía de Estilos) es un esquema relajado, sin colores muy artificiosos y que dejan distinguir bien las diferentes partes de la interfaz.

Mockups

Para poder llevar a cabo este apartado hemos hecho uso de mockups para que nos ayude a plasmar tanto las ideas funcionales que concebimos al principio como un diseño que lleve a cabo lo antes mencionado, y aunque está sujeto a cambios por temas técnicos, de conocimiento o por simple mejora, nos ayuda a dejar claras ciertas pantallas de forma que sea más rápido y eficiente la posterior codificación.

Para esto hemos usado la aplicación web Balsamiq [21], la cual nos aporta todo lo que necesitábamos para construir un primer diseño.

También, nos ayuda a despejar dudas de concepto, por lo que con unos buenos mockups iniciales podemos conseguir ver cosas que no veríamos sin ellos como por ejemplo las proporciones de cada componente, las transiciones entre pantallas... de forma que consigamos un diseño en que llevar a cabo una acción no necesite muchos pasos y así llegar a tener una sencillez que ayude al usuario en temas de accesibilidad y usabilidad.

Splash o pantalla de carga inicial:

Esta pantalla es una pantalla de transición desde el momento que el usuario pulsa sobre el icono de la aplicación hasta que puede usarla una vez se ha abierto la pantalla principal. Todas las aplicaciones poseen una ya que es una manera elegante de iniciar nuestra aplicación y nos va a servir en un primer momento para presentarnos al usuario ya que mostraremos una pantalla sencilla con los colores que usamos como parte de nuestra guia de estilos y un icono que elijamos para que sea nuestra imagen y distintivo.

Por otro lado, para cargar y pedir permisos de geolocalización. Con esta pantalla lo que buscamos es que, además de presentarnos al usuario, podamos terminar de cargar ciertos procesos y generar ciertos valores que necesitaremos para que la aplicación funcione correctamente de forma transparente al usuario. Así la persona que lo utiliza solo ve una pantalla con un diseño sencillo mientras la aplicación acaba de iniciarse.

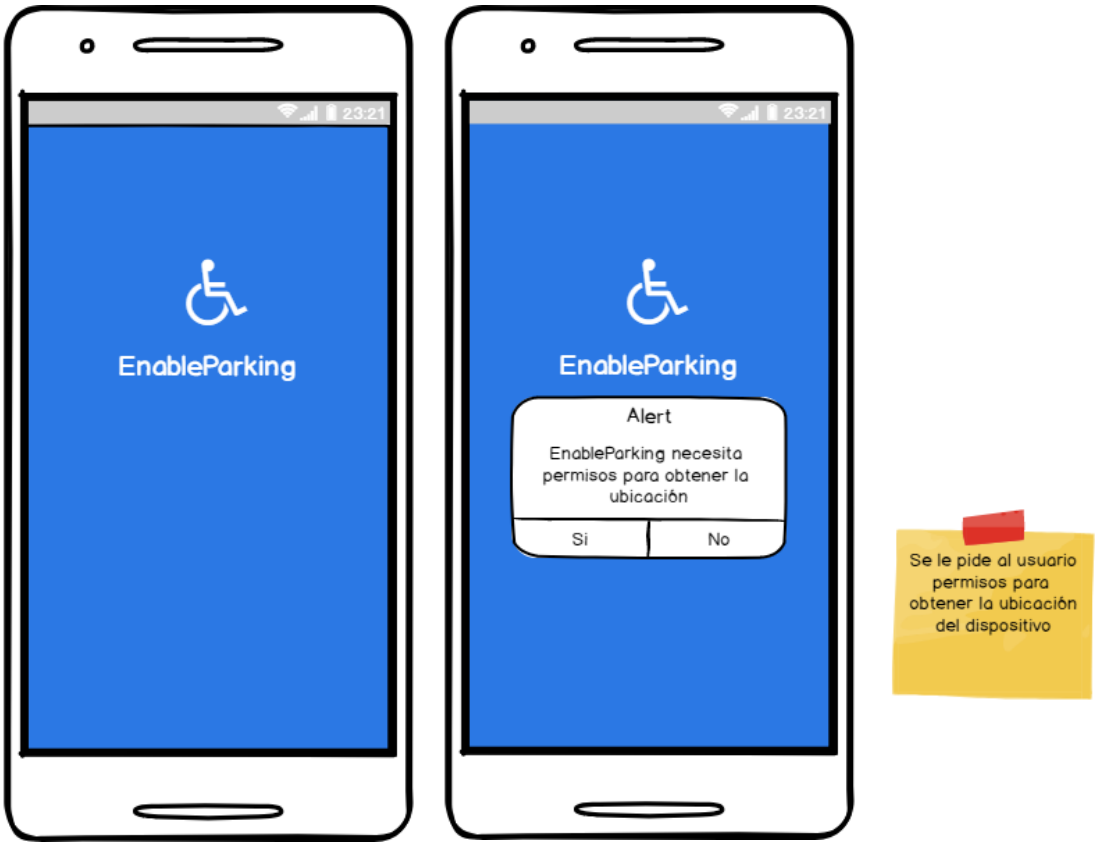

Figura 13 – Mockup 1: Splash screen

Esta pantalla haría referencia a nuestra especificación **RF 01** donde pedimos permisos al usuario para que la aplicación pueda acceder a la ubicación

Pantalla principal:

En cuanto a esta pantalla decir que es la que define la aplicación, es decir, la pantalla que posee casi todas las funcionalidades esenciales para el usuario. Es nuestra la sala principal donde ofrecer lo que el usuario necesitará al usar nuestra aplicación.

En la pantalla principal tendremos un mapa de donde podremos ver la ubicación actual del dispositivo que estemos utilizando y las marcas (chinchetas o markers) que nos indicarán donde podremos encontrar plazas de aparcamiento para personas con problema de movilidad.

De la misma forma, tendremos ubicado en la zona superior un buscador, donde pulsando se nos abrirá un teclado para introducir un lugar, una calle... Además, este buscador tendrá una lista de sugerencias de lugares y calles relacionados con lo que vamos escribiendo en la barra.

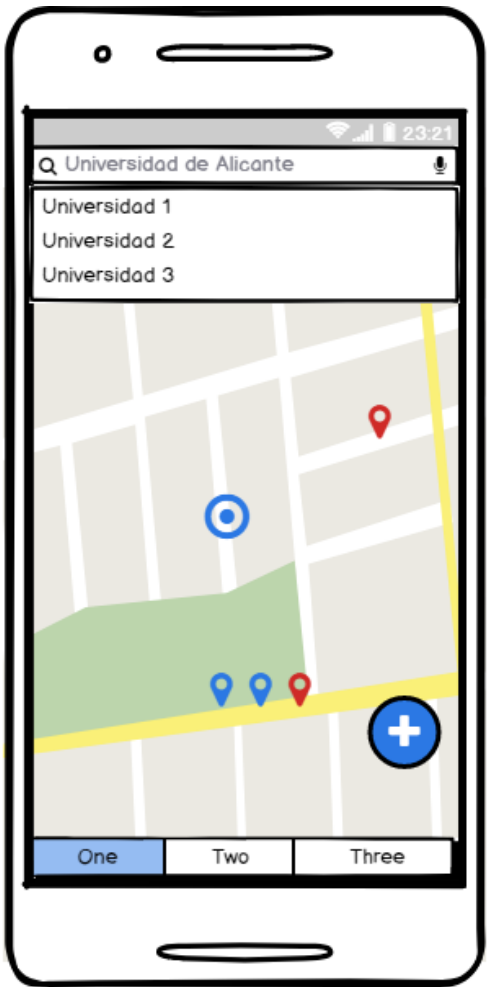

Figura 14- Mockup 2: Main Screen

En la parte inferior tendremos una barra/menú con varios botones:

- Botón 1: (One) Muestra o esconde la barra de búsqueda.
- Botón 2: (Two) Centrará la pantalla en nuestra ubicación.
- Botón 3: (Three) Nos dará la opción de sugerir una nueva plaza de aparcamiento.

En cuanto a la barra de búsqueda procuraremos que sea una búsqueda guiada con la cual podamos mostrar al usuario una lista de sugerencias mientras escribe de forma que podamos facilitar algo el encontrar el lugar al que se dirige.

Por otro lado, el botón que centra la pantalla es muy usado en todas las aplicaciones que contienen un componente de mapas, ya que podremos movernos libremente por el mapa y sea fácil volver mediante una transición al punto en el que se encuentra el usuario.

Por último, con el tercer botón vamos a poder implementar una funcionalidad más interactiva por parte de los usuarios de esta aplicación. Ya que no tenemos una base de datos pública a la que acudir para almacenar todas las plazas de aparcamiento para personas discapacitadas con este botón lo que buscaremos es que los interesados puedan mandarnos al sistema Firebase su ubicación indicándonos que ahí existe una plaza de aparcamiento que no tenemos registrada o no tenemos reflejada correctamente, de forma que la persona encargada de la aplicación (admón..) o un posterior sistema de procesado de datos pueda crear nuevas plazas de aparcamiento en la base de datos de EnableParking.

Figura 15 – Mockup 3: Pantalla con pop-up de sugerencias

También podremos ver un botón/menú flotante. Este menú flotante se desplegará una vez lo pulsamos y nos dejará elegir entre un mapa con estética de plano y una con estética de

satélite, que es un mapa con el aspecto de fotografías o satélite más realista. Esto puede ayudar cuando una zona nos es familiar de forma que de un vistazo sepamos en la zona en la que nos encontramos.

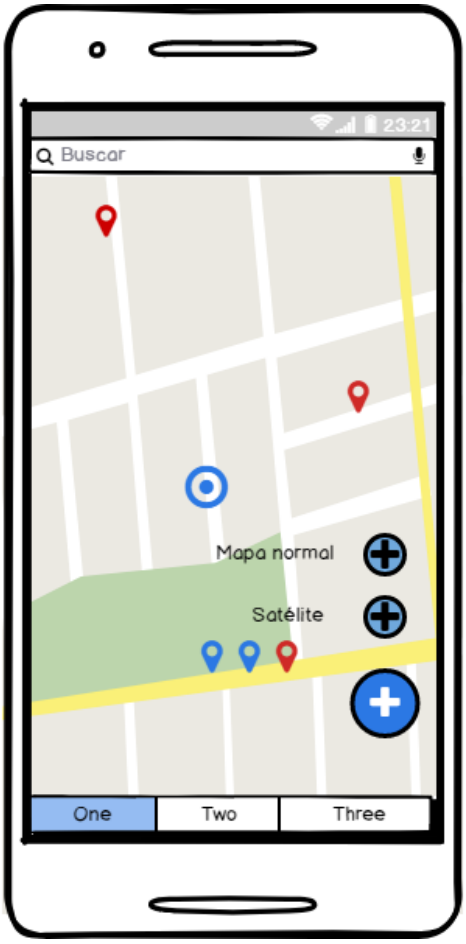

Figura 16 – Mockup 4: Botón cambio de mapa

En cuanto a los marker o chinchetas que marcan la ubicación de las plazas de aparcamiento serán pulsables, de forma que nos lleven a una pantalla personalizada de cada una de las plazas de aparcamiento donde podremos ver mediante una interfaz sencilla y con buen tamaño a qué plaza nos referimos y datos sobre ella.

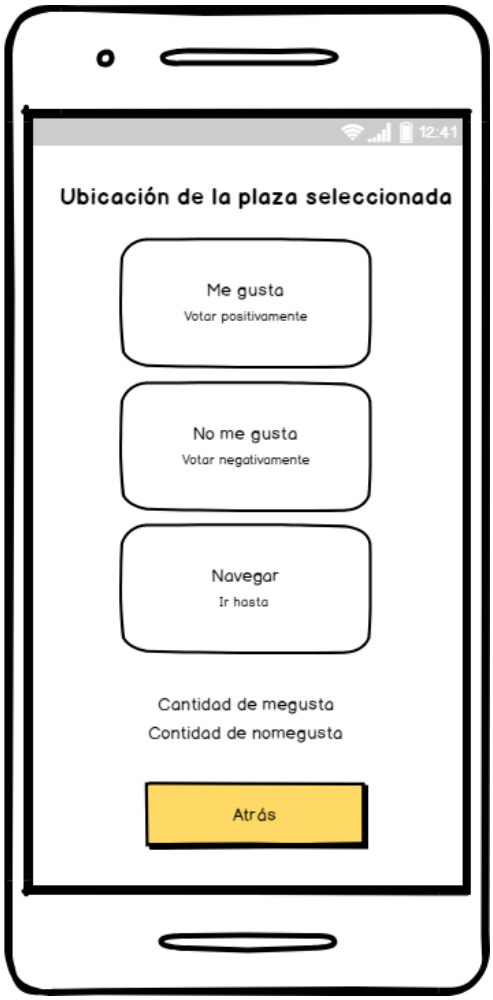

Figura 17 – Mockup 5: Detalle plaza aparcamiento

Primeramente, la calle y el número sobre el que se encuentra, como aproximación y punto de referencia. Después tendremos tres botones grandes e interactivos. Dos de ellos para puntuar la plaza de aparcamiento y un tercero para unas de las funcionalidades más importantes de la aplicación: navegar hasta la plaza indicada. Para esto hemos hecho uso de la aplicación de Google Maps que poseemos todos en nuestro teléfono móvil, de forma que al pulsar el botón se abra y nos guíe.

Para finalizar, además de mostrar la puntuación de "me gusta" / "no me gusta" de la plaza de aparcamiento, tendremos un botón para volver a la pantalla principal.

c. Guía de estilos

En este apartado vamos a ver la guía de estilos que se utilizarán en el proyecto.

Esto significa que todo el diseño de la aplicación girará alrededor del estilo que hemos decidido con anterioridad. En cuanto a la gama de colores e iconos nos hemos basado en los principios de Material Design [22].

Por una parte, los colores que hemos escogido han sido unos colores fáciles de complementar y que pensamos para que por el uso de la aplicación no cansen además de una paleta amplia para cubrir las necesidades que teníamos. Para generar la paleta hemos usado una aplicación web denominada Coolors [23] y este es el resultado:

Figura 18 – Paleta de colores EnableParking (Coolors)

Por otra parte, los iconos han sido extraídos de varias fuentes. La principal ha sido Android Studio, que pone a nuestra disposición un librería amplia y versátil, pero en otros casos donde pensamos que queríamos algo más atractivo visualmente hemos acudido a librerías web donde podemos usar varias de las creaciones que la comunidad pone a nuestra disposición de manera desinteresada como por ejemplo flaticon [24], Noun Project [25] y Map Icons [26], entre otras.

8. Implementación

En este bloque vamos a ver brevemente las estructuras tanto del frontend como del backend, veremos el control de versiones que, aunque es proyecto pequeño y de desarrollo individual, siempre es recomendable utilizar alguno y etapas del desarrollo ligadas a la metodología elegida.

En cuanto a la metodología nos decantamos por SCRUM, primeramente, porque es un desarrollo pequeño, que puede ir escalando poco a poco, pero que queremos conseguir un prototipo lo antes posible para presentar tanto el equipo de artefactos como a las personas de la asociación AMFI que son los principales interesados.
Además, es un equipo de una persona, donde tanto el backend como el frontend y el diseño UX está llevado a cabo unilateralmente (aunque siempre se pueden hacer cambios por consejo de mi tutor) y la organización puede ser más flexible.

a. Estructura del frontend

En este apartado vamos a ver un pequeño resumen de la estructura frontend que han sido utilizadas para crear EnableParking.

Como ya hemos visto antes la tecnología utilizada para desarrollar la aplicación móvil ha sido Android en su vertiente de Java, que es donde más experiencia se tenía gracias al Máster y a la carrera, por lo que el aprendizaje a priori era menos pronunciado que el que habría que haber tenido si nos hubiéramos decantado por otra plataforma. Además de que me sentía más cómodo con las herramientas como Android Studio, git… que tanto hemos usado a lo largo de este Máster.

Si bien es cierto que me gustaría poder desarrollar la parte frontend en alguna tecnología híbrida como pueda ser Ionic 4, React o Vue.js entre otros frameworks, ya que, aunque no son nativas el rendimiento nativo no es tan necesario para esta solución, puesto que no es muy pesada.

Decir que Android es un lenguaje donde el patrón que se sigue es el de Modelo-Vista-Controlador (MVC).

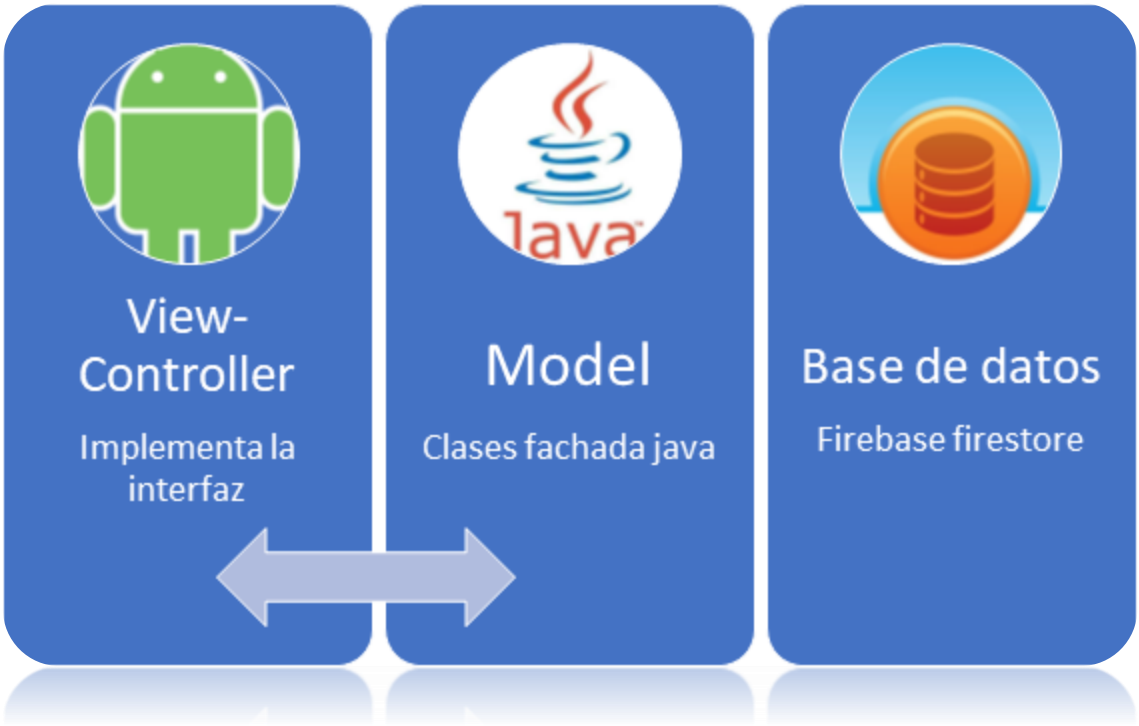

Figura 19 – Capas del sistema

Un ejemplo básico para ejemplificar esto es que, dentro del proyecto de Android, que creamos en nuestro Android Studio, contiene varios tipos de archivos, pero básicamente nos centraremos en dos.

Por un lado, tenemos las clases que utilizamos como modelo que en nuestro caso son los ficheros JAVA los que forman los objetos que representan las plazas de aparcamiento con todos sus atributos (estado, coordenadas, calle...), getters y setters. A estas clases muchas veces se les asocia clases controlador que están en una capa intermedia entre el modelo y la vista. Pero como vemos en el diagrama anterior en nuestro caso no es así.

En cuanto a la parte de la vista las actividades (activity), (que llevan ligadas normalmente un fichero XML de estilo y maquetación) son las encargadas de mostrar por pantalla los datos además de gestionar también las interacciones que el usuario lleva a cabo en el dispositivo. En nuestro caso al ser un "modelo activo" (**active model**) estas clases activity también pueden actuar como controlador e interactuar con el modelo, el cual hemos implementado como una fachada en JAVA de nuestra base de datos en la nube.

Estructura de actividades

En este subapartado vamos a ver en qué actividades se compone nuestro proyecto Android, estas son las entidades o componentes básicos que contienen las pantallas con las que el usuario podrá interactuar para llevar a cabo acciones como por ejemplo usar un mapa, hacer una llamada o mandar un email, entre otras [27].

Cada actividad tiene asociada una ventana en la que se puede dibujar su interfaz de usuario, como hemos dicho antes, la ventana generalmente abarca toda la pantalla del dispositivo.

Las aplicaciones normalmente se componen de una o varias actividades que se relacionan entre sí. Siempre tendremos una principal y desde esta podremos acceder a las demás mediante acciones, que hará que dejemos de ver una pantalla para comenzar a ver otra que pasará a la pila de actividades que maneja nuestra aplicación.

En nuestro caso la actividad principal es una pantalla compuesta por un mapa, un buscador y varios botones mediante lo que podremos hacer diferentes acciones. Algunos abrirán nuevas actividades a las que transicional y otros no.

La primera de nuestras actividades es la SplashActivity la cual es una actividad de transición, la utilizamos como presentación y para pedir los permisos de ubicación de forma temprana si por lo que sea el dispositivo del usuario no navega con fluidez a la actividad principal.

Toda nuestra aplicación está diseñada alrededor de la main activity ya que como vimos en el estudio es la pantalla que más uso va a darle el usuario, en nuestro caso es una actividad formada por un mapa con varios botones y una barra de búsqueda.

En esta actividad controlamos que poseamos los permisos necesarios para acceder a la ubicación del dispositivo.

SplashActivity:

En un principio cuando instalamos la aplicación mostramos un mensaje emergente o popup demandando permiso y damos dos opciones. Si el usuario acepta utilizamos el servicio que Google pone a nuestro servicio para pintar el mapa y pintar la ubicación en tiempo real, por el contrario, si se deniega la petición desaparecerán la botonera y aparecerá un banner en su lugar para que el usuario de los permisos de ubicación cuando crea pertinente.

MainActivity / MapsActivity:

El primer trabajo que lleva a cabo esta actividad cuando llegamos a ella es preparar el mapa, el menú, la barra de búsqueda, etc.

Desde esta actividad podemos llevar a cabo diversas acciones. Por un lado, en el menú tenemos 3 botones, como explicamos anteriormente, todos controlados en esta actividad. Además, estas acciones no llevan a cabo una transición hacia otra actividad.

En cuanto a la barra de búsqueda la implementación es más compleja ya que posee una lista de sugerencias que va cambiando conforme vamos escribiendo en ella. Este componente lo hemos añadido con una librería compleja, materialSearchBar [28], que además utiliza diversos datos de la ubicación del usuario como por ejemplo dónde se encuentra o qué tipo de filtro debe aplicar para buscar en el API de Google Maps, como por ejemplo ADRESS con el cual conseguimos que busque por calles.

Cuando todos los elementos están cargados y preparados la actividad está preparada para pintar en el mapa los markers que representan las plazas de aparcamiento almacenadas en nuestra base de datos en la nube. En este caso lo que hace nuestra actividad es hacer una llamada mediante un SnapshotListener [29].

Con esto lo que conseguimos es crear un procedimiento mediante el cual estamos escuchando pasivamente nuestra base de datos, pero filtrando (haciendo una foto) tan solo a lo que nos interesa. Puede ser un documento, puede ser una colección completa o alguna combinación de ellas. De esta forma lo que conseguimos es recibir inmediatamente cualquier cambio que se de en la base de datos dentro de nuestro filtro.

En este caso hemos filtrado toda la colección de documentos que representan a las plazas de aparcamiento. De forma que cuando el estado de alguna se modifique, mediante los dispositivos hardware que irán instalados en cada una de las plazas de aparcamiento, podamos actualizar el color de cada marker en tiempo real.

ParkingActivity:

Cuando pulsamos en el infoWindow de un marker nuestra aplicación lo que hace es mediante un intent crea una transición de la actividad principal a esta actividad (enviando ID de la plaza entre intents) con la que estamos interactuando.

Esta actividad es la pantalla del detalle de cada plaza de aparcamiento y se rellena haciendo una consulta a nuestra base de datos, pero esta vez dirigida a un documento en concreto mediante un método en el cual aplicamos al filtro el ID de plaza y solo pintamos cuando la llamada se realiza correctamente (isSuccessful).

Una vez tenemos todos los campos rellenos ya podemos utilizar todos los botones. Los dos primeros son para que el usuario vote positiva o negativamente una plaza de aparcamiento según su opinión y el último botón sirve para navegar desde nuestra posición hasta la plaza de aparcamiento.

Esta parte la hemos implementado para que cuando el usuario pulse el botón de navegación nuestra aplicación abra Google Maps de forma que se le pase la ubicación de la plaza y el usuario pueda usar una aplicación de un proveedor donde conoce el servicio.

Flujo de las actividades

Para explicarlo un poco mejor vamos a ver el flujo de las actividades en nuestra aplicación. Aquí veremos los casos de uso posibles que el usuario podrá trazar una vez inicie EnableParking.

Estos casos de uso o caminos los vamos a representar mediante unas figuras y estas mostrarán un diagrama de flujo. En estos diagramas podremos ver las actividades/mockups representados por cuadrados que albergarán la acción acompañada del identificador del mockup correspondiente en el que se puede realizar (que hemos creado y adjuntado con anterioridad en el punto 7.2. Diseño de interfaces, apartado mockups), unidos por líneas que serán los caminos seguidos por el usuario al hacer uso de las distintas acciones disponibles.

Para llevar a cabo estos diagramas hemos buscado aplicaciones populares para hacer diagramas de flujo y finalmente nos hemos decantado por draw.io [30] que es una aplicación gratuita, que no requiere login y que además tiene gran cantidad de elementos que nos ayudan a expresar lo que deseamos.

A continuación, vamos a ver un diagrama de un caso de uso (UC) básico en el cual un usuario entra en nuestra aplicación. Nos sirve para ver un poco cómo se van a componer los siguientes.

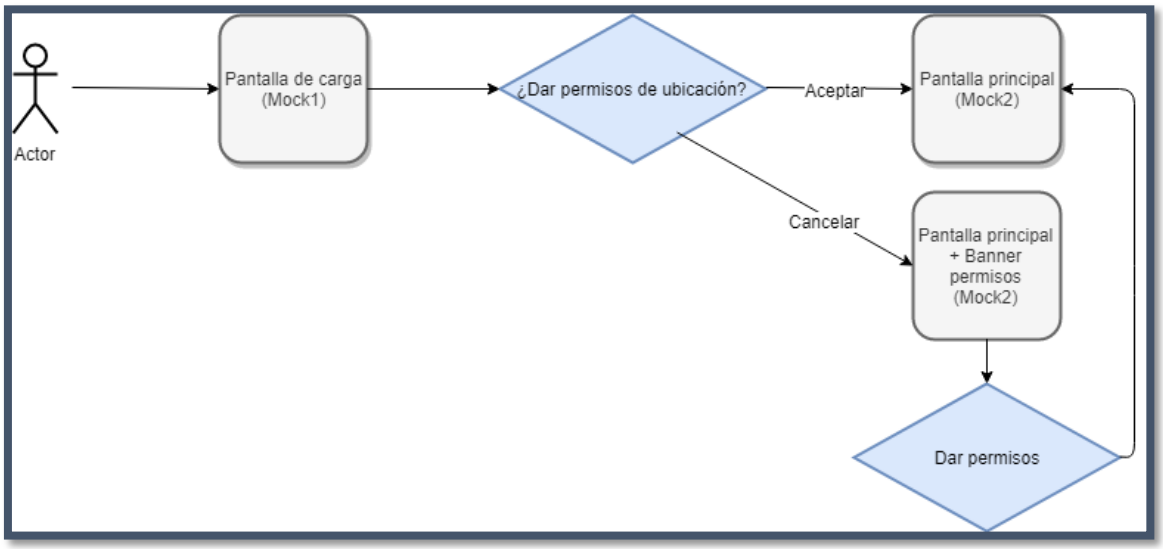

Figura 20 – UC 1: Desde la pantalla de carga

A continuación, vamos a ver los caminos y acciones que el usuario puede seguir una vez está en la actividad principal del mapa:

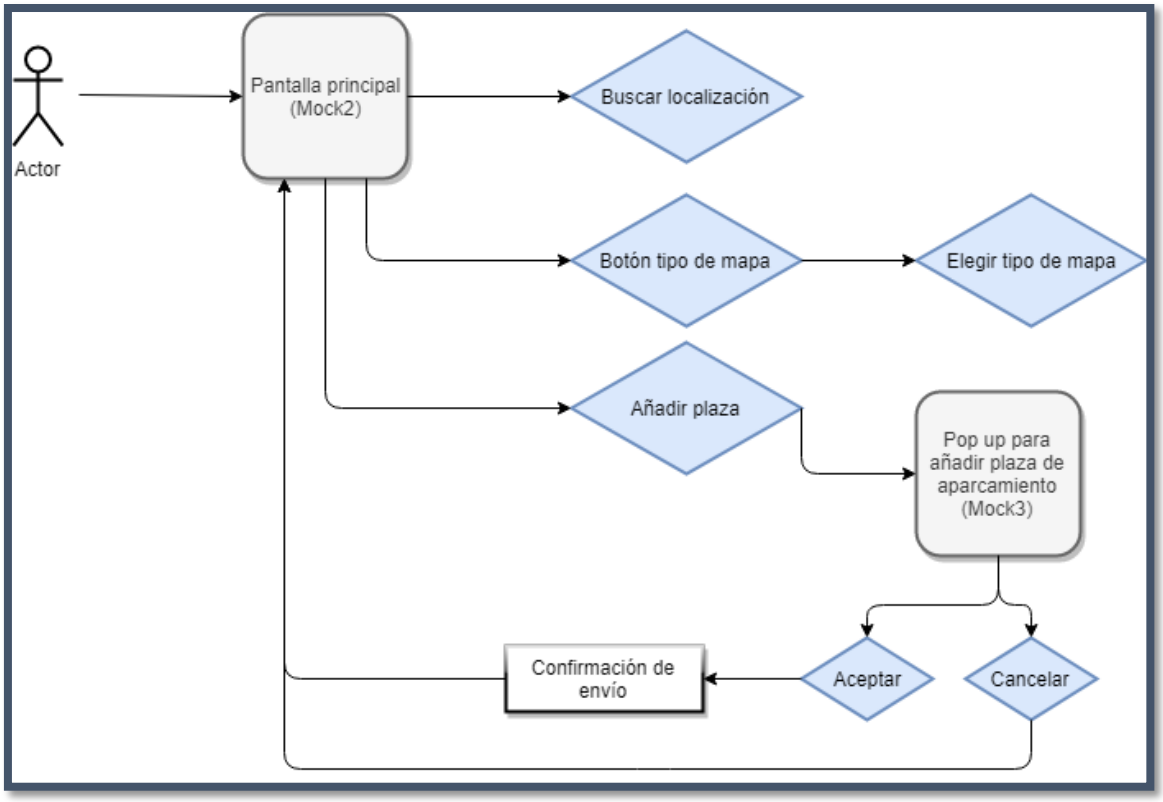

Figura 21 – UC2: Pantalla principal con menú

Por último, vamos a ver el UC al que llega un usuario al pulsar sobre el infowindow de una plaza de aparcamiento.

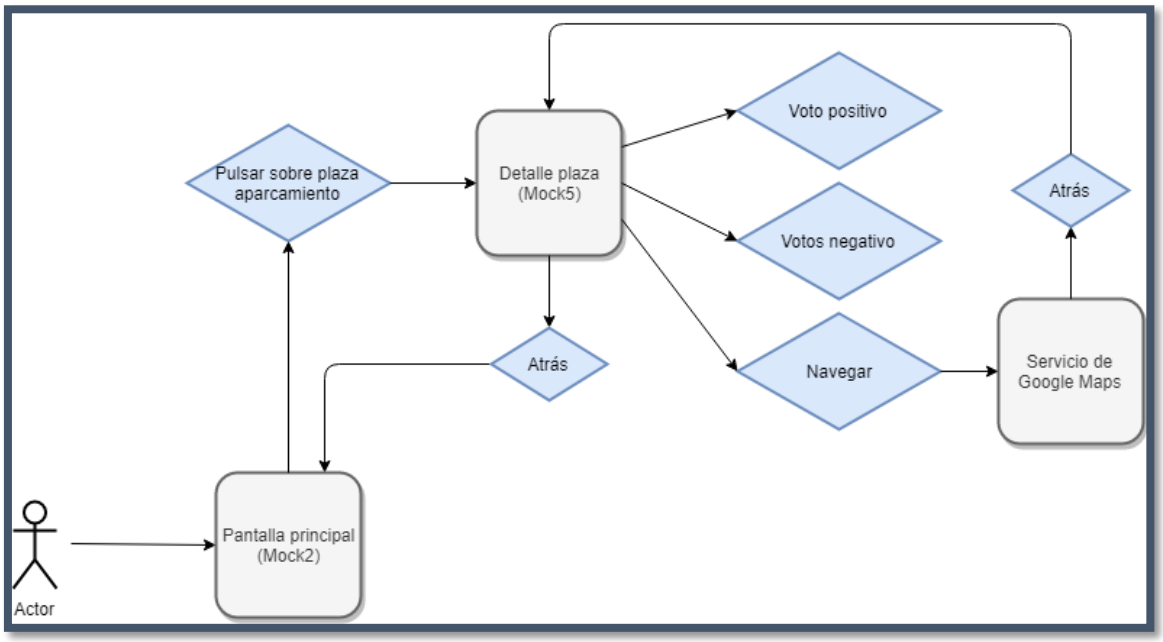

Figura 22 – UC3: Detalle plaza de aparcamiento

b. Estructura del backend

Aquí vamos a ver brevemente cómo hemos implementado el backend de nuestra aplicación ya que anteriormente hemos visto cómo ha sido el diseño de las estructuras de datos utilizadas.

Nuestro backend está compuesto por una base de datos en la nube, en este caso nuestro proveedor es Google y su servicio de Firestore, en nuestro caso la modalidad Cloud firestore, del cual ya hemos explicado antes cómo es su estructura de datos.

Las razones por las que hemos escogido Firebase Cloud Firestore [31] son diversas, pero las características más interesantes que veo a la hora de implementar este tipo de base de datos es que tenemos un almacenamiento en la nube, esto significa que la persistencia que conseguimos nos ayuda a no tener que invertir mucho en infraestructuras y más si la solución que implementamos no requiere un plan contratado muy alto.

Por otro lado, tenemos la seguridad de que la pérdida de datos por parte de un error de hardware la tenemos resuelta ya que iniciando sesión en un navegador podemos gestionar nuestro negocio desde cualquier sitio, con lo que conseguimos fiabilidad y efectividad si no necesitamos ningún motor de base de datos más potente y relacional.

Y finalmente hay que destacar que esta tecnología es fácilmente implementable en multitud de proyectos de diversos lenguajes gracias a que Google da soporte a todas o casi todas, por lo que podríamos servir a varias aplicaciones si fuese necesario de forma fácil y abierta.

Además, no conocía la tecnología lo que me motivaba a conocerla y aprender a utilizarla, además de venir muy bien para esta aplicación que necesitaba una base de datos pequeña.

Ahora bien, la forma de implementar Firestore con Android Studio integrar el SDK de Cloud Firestore en nuestro Gradle, que es el fichero de dependencias del proyecto Android.

De esta forma en la parte de la vista (MVC) podemos crear unos agentes de escucha en tiempo real dirigidos a alguna de nuestras colecciones en la base de datos para recuperar los datos.

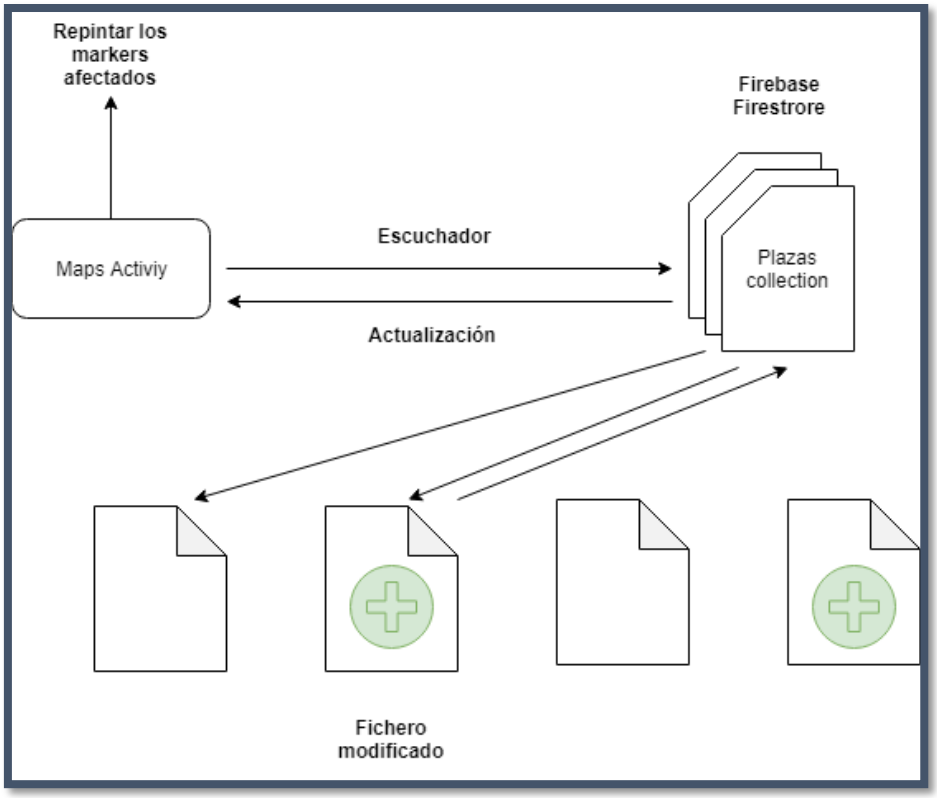

Figura 23 – Diagrama de actualización de datos en tiempo real

Cuando alguna de estas es modificada obtenemos estos cambios y los plasmamos para que el usuario tenga actualizado en cada momento con la versión más reciente. Esto lo hacemos por ejemplo cuando queremos todos los documentos de una colección, como se hace cuando queremos pintar todos los markers en nuestro mapa. Aunque normalmente se podría filtrar para que solo nos diera los de la zona en la que se encuentra buscando el usuario si fuese necesario por temas de rendimiento.

Además, también podemos crear consultas expresivas, las cuales podemos crear para recuperar documentos individuales como hacemos cuando buscamos el detalle concreto de una plaza de aparcamiento.

De la misma forma podemos instanciar objetos y enviarlos para crear nuevas colecciones, documentos o modificar los que ya existiesen teniendo la referencia al mismo.

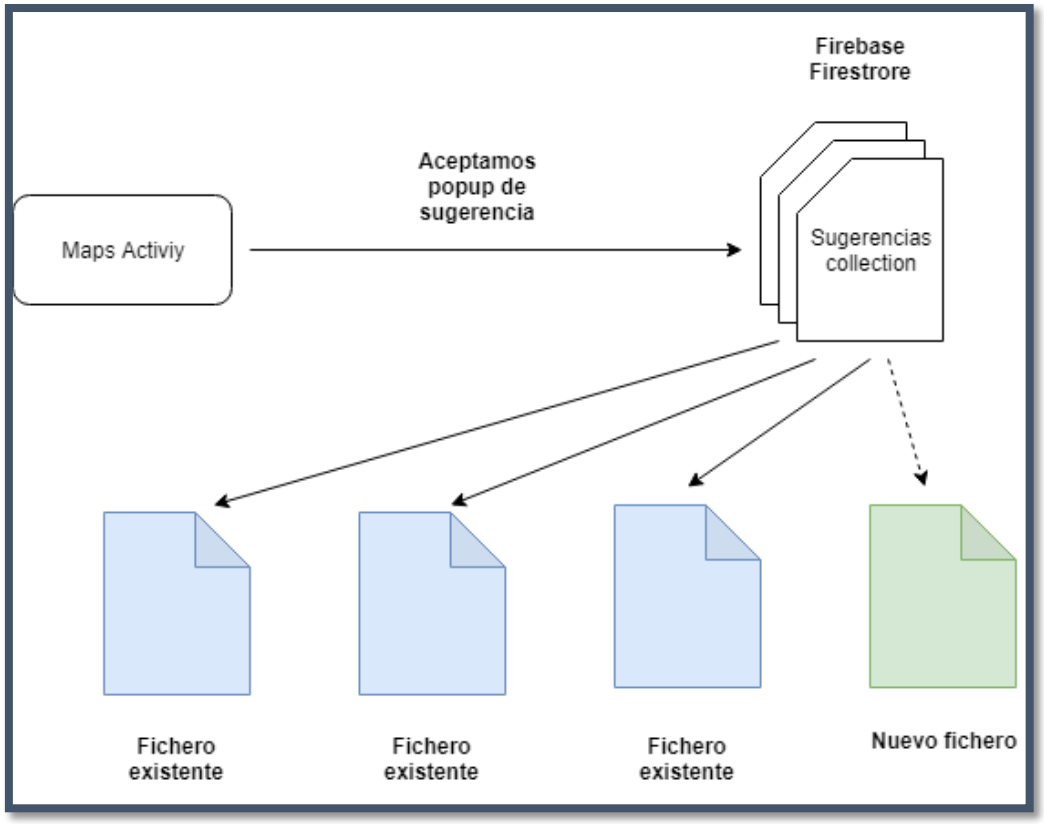

Figura 24- Diagrama de creación de nuevas plazas

Como por ejemplo el mecanismo que tenemos para que los usuarios puedan sugerir nuevas plazas de aparcamiento.

Ahora vamos a ver cómo hacemos para votar una plaza de aparcamiento o cómo hacemos en el caso de que una plaza ya no exista y los usuarios quieran darnos este feedback.

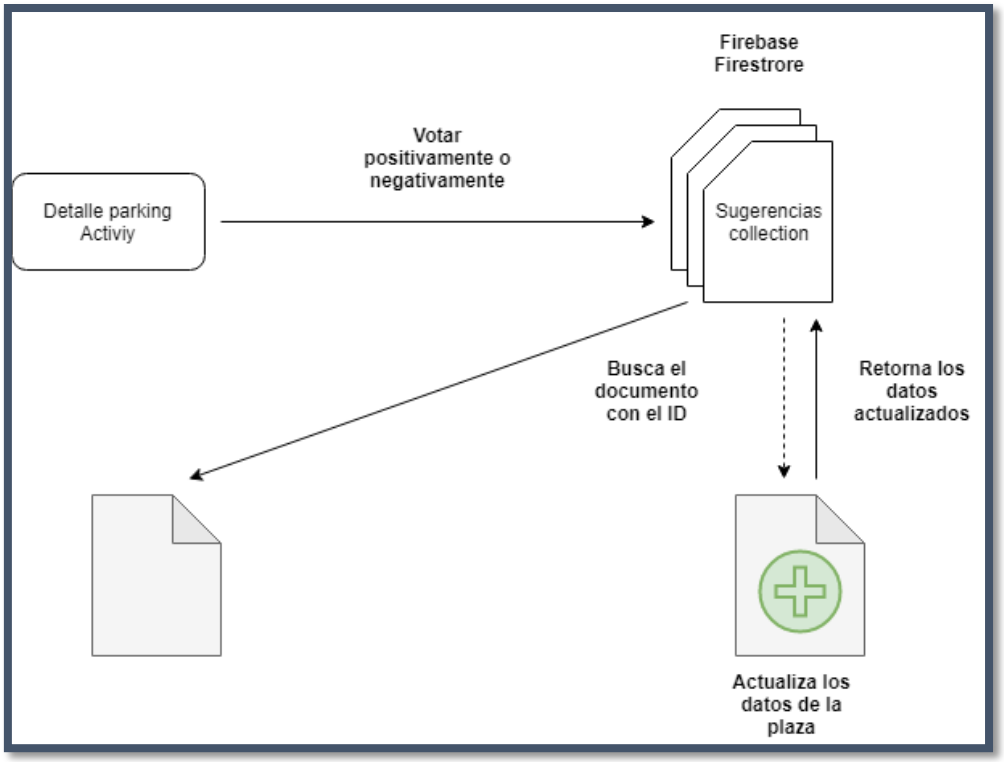

Figura 25 – Diagrama: cómo votar una plaza

Como podemos ver en este diagrama cuando actualizamos una plaza lo que hacemos es buscar mediante el ID que tiene el documento. Una vez actualizamos los datos en la BD recuperamos el documento actualizado para volver a pintar automáticamente en la pantalla del usuario y en la de todos los usuarios.

Del mismo modo lo podemos hacer el mandar una advertencia de que la plaza ha dejado de existir. Mandamos el ID y creamos un documento con el identificador, ya que es más fácil de ver para la persona que administre el sistema que crear un campo en algún documento ya existente.

En caso de que recibamos mas de una advertencia de una plaza concreta lo que hacemos es crear un contador de veces que ha ocurrido, con lo que podemos contabilizar. En caso de que el número sea relevante podríamos ver si esto es cierto y actuar en consecuencia.

c. Sistema de control de versiones

En este pequeño apartado simplemente queremos dar énfasis a la importancia de tener una buena herramienta de control de versiones. Y esto no es solo por lo que haya podido estudiar en la carrera o lo que me hayan podido contar, sino porque la experiencia que he recabado no solo en mi carrera profesional si no también hace unos años cuando realicé mi TFG me han hecho verlo.

En aquel de mi TFG el disco duro de mi portátil dejó de funcionar y solo puede recuperar ciertas cosas que tenía almacenadas y duplicadas en otro sistema de almacenamiento, por lo que en este caso hemos subsanado el error y hemos usado una herramienta como Git.

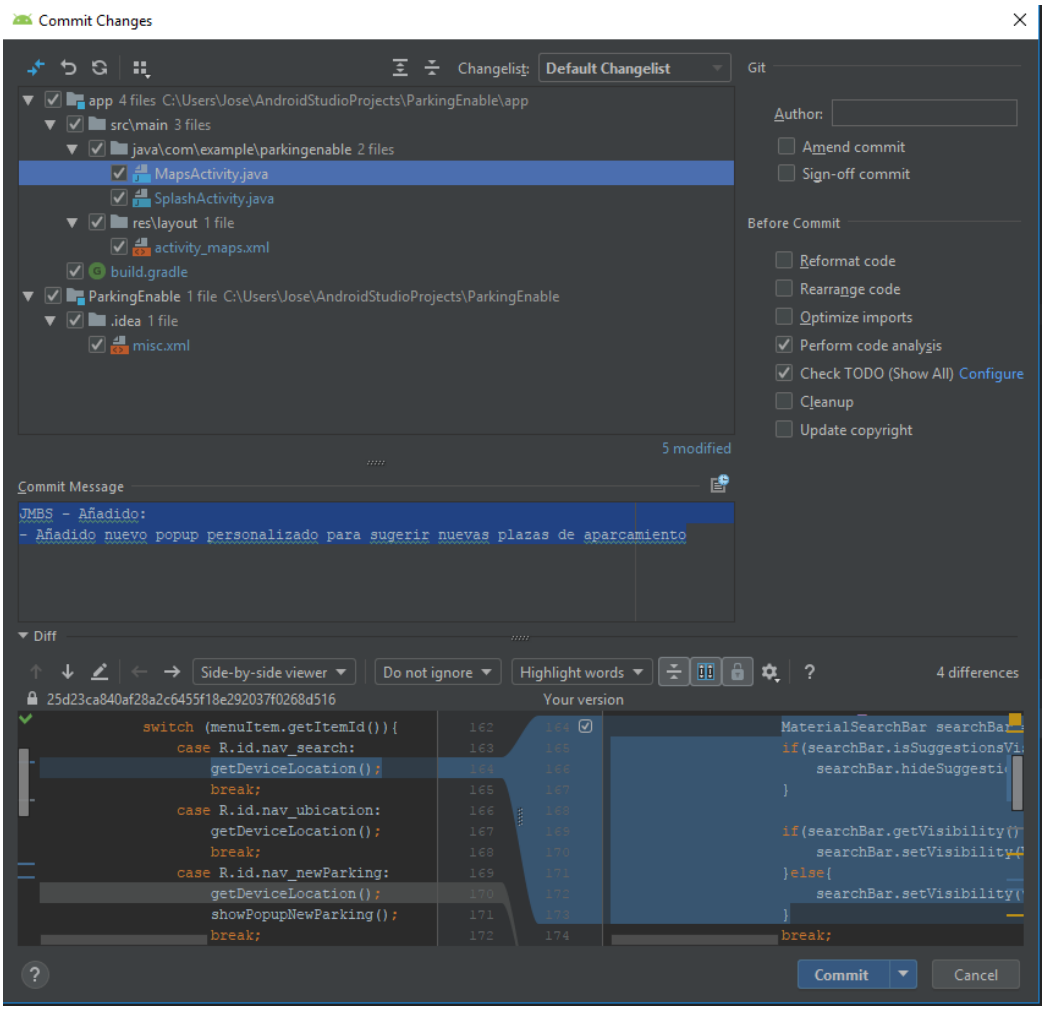

Figura 26 – Ejemplo subida a Git

Git es una de las herramientas más potentes de control de versiones y que gracias al máster lo he podido implementar con cierta facilidad dentro del IDE Android Studio como si de una extensión se tratase.

Por lo que el trabajo rutinario de subir a la nube mis cambios y deshacerlos en caso de tener problemas de implementación ha sido mucho más fácil e intuitiva gracias a los comparadores de documentos que pone a nuestra disposición.

d. Etapas del desarrollo

Cómo comentamos al inicio del apartado nos decantamos por una metodología ágil porque es un proyecto ligero en el que podíamos hablar activamente con la asociación AMFI, que podríamos considerar el cliente, y con nuestros compañeros de Artefactos de la UA por lo que al recabar información y diseñar unos requisitos mínimos pudimos trazar las primeras tareas de desarrollo, añadiendo posteriormente las tareas de documentación, aprendizaje…

Por lo que pudimos dividir todas las tareas en dos sprints.

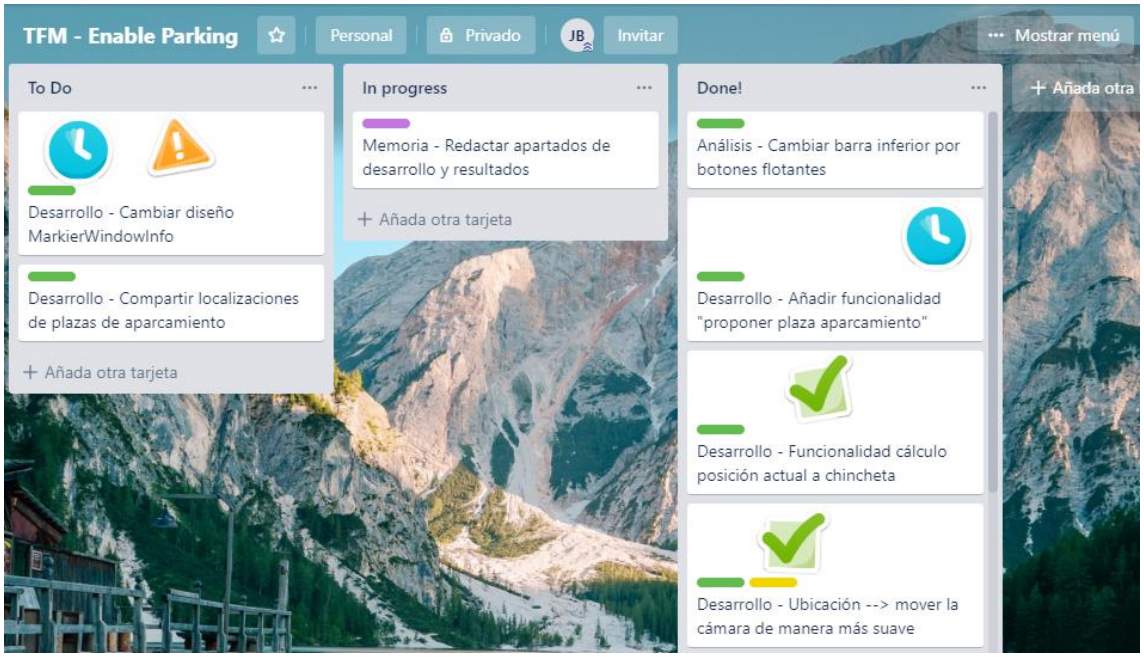

Figura 27 – Tablero SCRUM avanzado en el tiempo

Hay que comentar que nosotros solo necesitamos tres columnas en nuestro tablero, pero suelen haber más columnas en los tableros o pizarras SCRUM cuando los proyectos son más grandes ya que puede que tengamos un equipo de calidad/testing que tiene las tareas en esa columna hasta que se dan por finalizadas las pruebas, de forma que pasarían a una última columna de validación o a otra anterior como señal de que se han encontrado fallos en la misma.

Para finalizar decir que creamos un primer sprint, de unas 5 semanas, con tareas tanto de desarrollo como de documentación previa. Primeramente, dedicamos los inicios del primer sprint para refrescar conocimiento de Android y posteriormente para conocer la tecnología de base de datos que íbamos a utilizar.

Una vez tuvimos lo anterior claro pasamos a darle forma al proyecto principal de forma que las funcionalidades más básicas estuvieran implementadas al finalizar el sprint, como pueden ser mostrar las plazas de aparcamiento en el mapa, obtener la ubicación de forma correcta del dispositivo del usuario, etc.

Y de cara al segundo sprint pasamos a dedicar más tiempo tanto a la barra de búsqueda con una lista de sugerencias, lo cual nos absorbió bastante más tiempo del esperado. Posteriormente modificamos y añadimos pantallas con más carga de trabajo en el diseño y por último funcionalidades secundarias que finalizarán al acabar este sprint en el que nos encontramos ya que gran parte de este sprint está copado con tareas de redacción de la memoria.

9. Resultado

El objetivo principal de este proyecto ya lo hemos comentado a lo largo del documento muchas veces y hemos visto también que la solución ha sido llevar a cabo la implementación de un sistema de monitorización, geolocalización y enrutamiento hacia las plazas de aparcamiento para todo aquel que use nuestra app.

Si bien es cierto que la parte hardware corre a cargo de los chicos de artefactos, hemos dejado la aplicación preparada para su uso a falta del envío de datos por parte de los dispositivos o balizas instaladas a posteriori.

En este apartado vamos a ver hasta dónde hemos llegado en el momento final del segundo sprint y por lo tanto el cierre de la memoria. Lo que no significa que el desarrollo de la aplicación y si posterior evolución no sea posible.

Tanto si hemos tenido problemas a lo largo de desarrollo como si no vamos a plasmarlo. Además, podremos comprobar si hemos llegado a los a los requisitos que consideramos mínimos para que la aplicación posea una versión estable con la que empezar a funcionar en un escenario real.

Además, vamos a exponer mediante capturas cómo ha quedado el resultado y así podremos también comparar con lo previsto en fases anteriores del desarrollo, como el análisis o el diseño de las pantallas o interfaces.

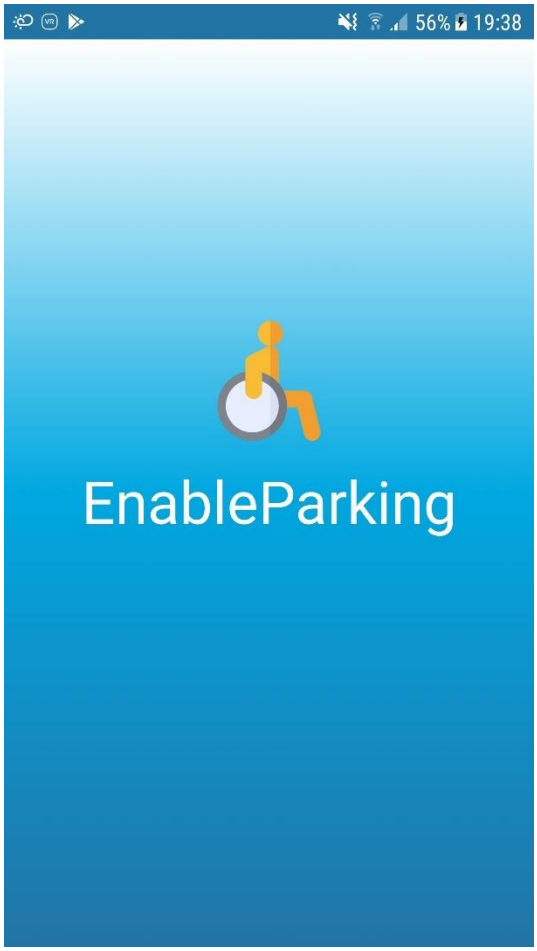

Figura 28 – Interfaz de pantalla de inicio

Esta es la interfaz de la splash activity, es la pantalla que la aplicación utiliza para cargar antes de mostrar la pantalla principal. Una vez que la aplicación es mantenida abierta por el sistema del móvil ya no se vuelve a ver hasta que el usuario mata la aplicación.

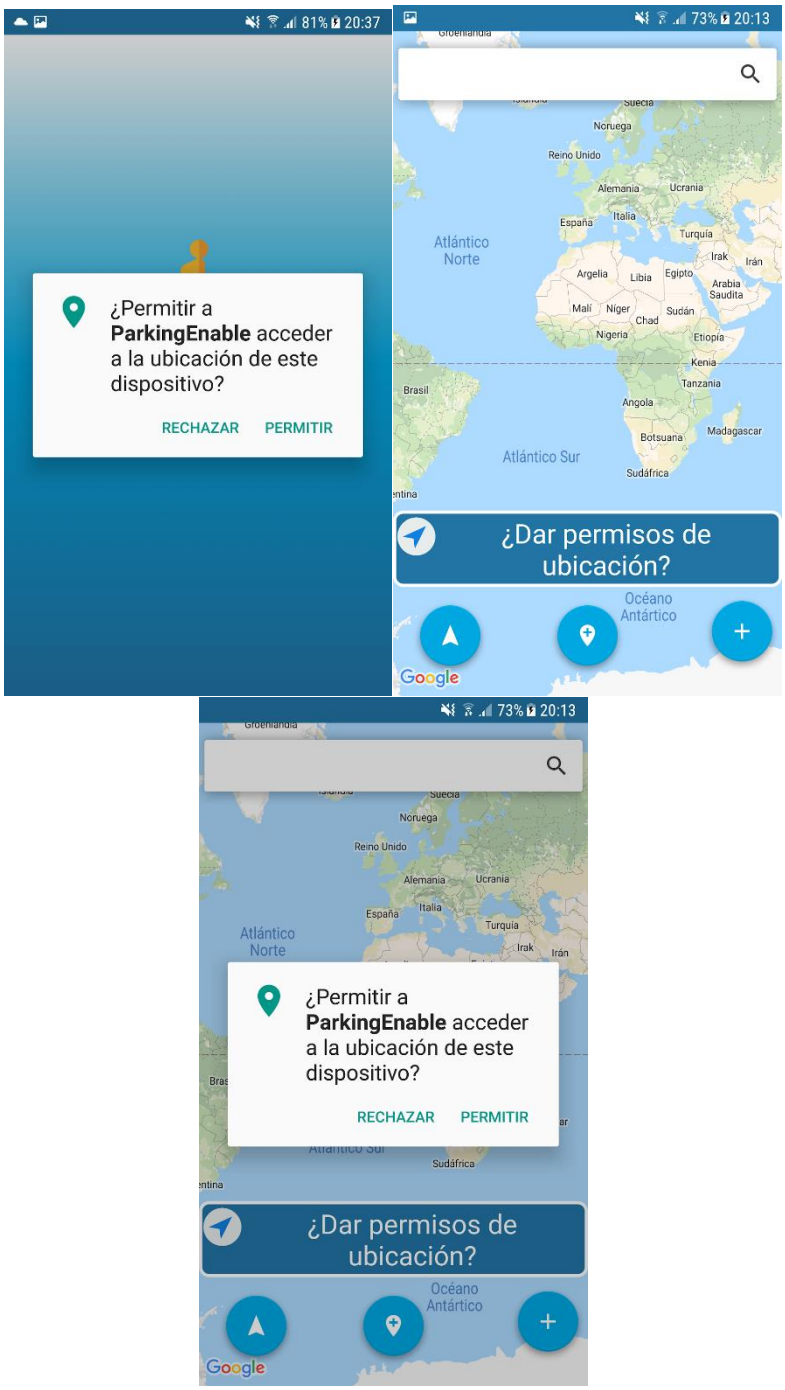

Figura 29 – Interfaces con petición de permisos

Aquí vemos como si no aceptamos los permisos durante la pantalla de carga, la primera vez que arrancamos la aplicación tras la instalación, tendremos un banner para pulsar en caso de que queramos que la aplicación funcione correctamente.

Al pulsar sobre el banner conseguimos abrir de nuevo el diálogo de forma que la aplicación adquiera todas sus funcionalidades al aceptarlo.

El banner fue un componente desarrollado personalmente para esta pantalla, pues hay casos en que las personas cuando utilizamos una aplicación denegamos que puedan saber nuestra ubicación.

Por ello pensamos en añadir un banner con el cual podamos dar fácilmente los permisos en el momento en que queremos. Lo que hace la actividad principal es preguntar si el usuario ya nos ha dado permisos y si no lo ha hecho muestra este mensaje que flota por encima del mapa.

Cuando el usuario lo pulsa y acepta lo que ocurre es que la aplicación esconde este componente.

Con esto conseguimos que ir a darle permisos a la aplicación no sea complicado, incluso para personas que no tienen porque saber cómo funciona el funcionamiento de sus aplicaciones en el sistema.

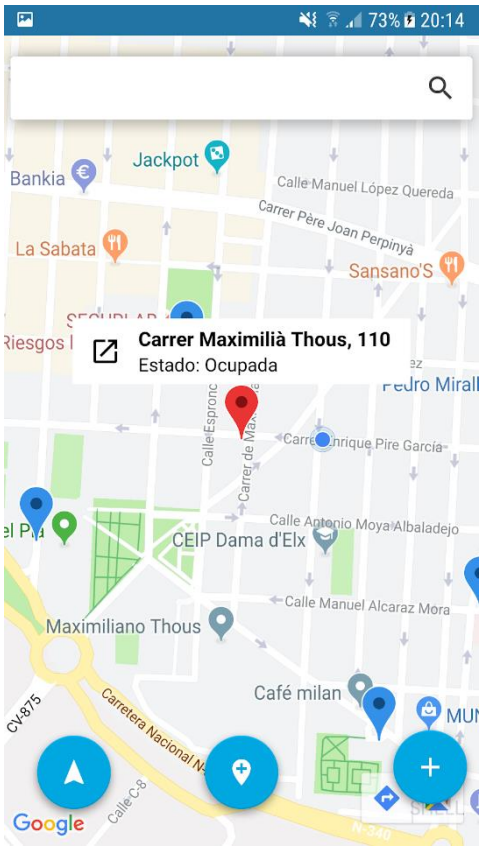

Figura 30 – Interfaz con mapa y markers personalizados

En esta figura podemos ver mucha información, aunque hemos intentado que todo quede lo más claro posible.

Por un lado, arriba tenemos la barra de búsqueda, que como veremos en otra figura sugiere direcciones al tiempo que escribimos, además de autocompletar.

Este componente ha tenido que ser creado con la ayuda de la comunidad. Nos hemos apoyado en buscadores con sugerencias que ya existían en github y los hemos adaptado.

Este componente trabaja directamente con la ubicación de nuestro usuario por lo que ayuda a que las sugerencias sean más acotadas y de esta forma no nos sugiera cosas que estén en otro país o continente con las primeras letras que escribamos.

Asimismo, decidimos que el buscador no se muestre o esconda cuando pulsásemos un botón de búsqueda, como teníamos al principio. Pues vimos que no molestaba y verlo en un primer momento ayuda al usuario a saber lo que está haciendo.

Hay que decir que programamos para que cuando pulsamos sobre la barra de búsqueda el teclado se despliegue y que usando pulsamos sobre el mapa se vuelva a esconder. Todo esto se controle en la actividad principal o **MapsActivity**

Por otro lado, vemos los markers que apuntan a las plazas de aparcamiento.

Importante ver que son de dos colores distintos. En el caso del azul (pensamos ponerlo en verde), significa que la plaza está libre y en caso de rojo significa que la plaza está ocupada como indica el infowindow personalizamos que se nos muestra al pulsar sobre cualquiera de ello.

Por otro lado, tenemos dos botones y un menú. Todo flotantes por encima del mapa:

- El primero es el que nos devuelve a nuestra posición en el mapa, es decir se centra en nosotros.
- El segundo es el que (como veremos en la imagen siguiente) nos da la opción de mandar una sugerencia de plaza de aparcamiento que no está reflejada en nuestra base de datos y por lo tanto en el mapa.

Y el tercero es un menú que nos dejará elegir entre un mapa como el que vemos y otro tipo satélite. Como veremos más adelante.

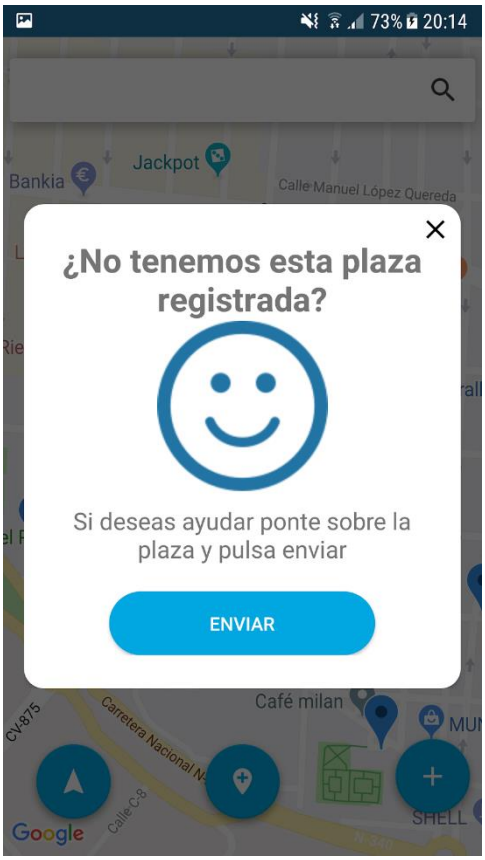

Figura 31 – Diálogo: sugerencia de nueva plaza

Aquí vemos qué ocurre cuando pulsamos sobre el botón central de la pantalla, el que posee el marcador con un símbolo más (+).

En esta figura vemos cómo hemos construido un diálogo personalizado ya que queríamos que la aplicación no quedara tan genérica con los diálogos predeterminados de Android Studio. Así pues, creamos un layout personalizado el cual utilizaremos de forma recurrente en toda la aplicación.

Esto nos permite replicarlo y modificarlo dependiendo de la situación dentro del flujo de acciones. Si bien es cierto que no lo usamos al principio cuando pedimos los permisos de ubicación, esto es porque dese Google no recomiendan que los diálogos de permisos pierdan su forma.

Con este mensaje conseguimos que sea sencillo mandar la ubicación de una plaza que no tenemos en base de datos al pulsar sobre el botón central de la pantalla y aceptando.

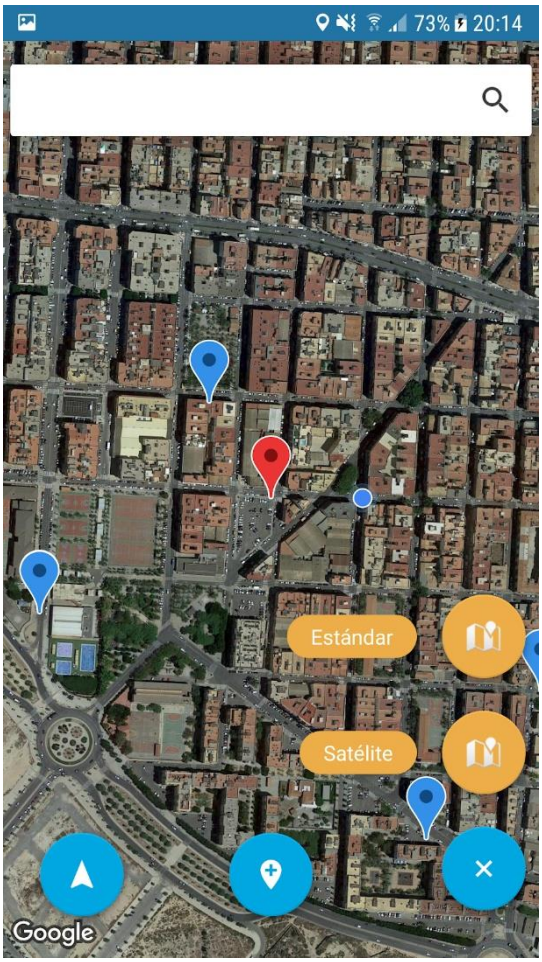

Figura 32 – Menú para cambiar el tipo de mapa

Esta es una funcionalidad sencilla, pero que ayuda mucho a orientarse cuando alguien busca lo que sea en un mapa. Con estos dos botones en el menú podremos ir alternando entre cualquiera de los dos tipos de mapa.

Esta funcionalidad no ha sido personalizada en exceso. Es cierto que hemos querido mantener la forma flotante de los demás botones, que en un principio fueron diseñados como una barra, pero esta funcionalidad nos la da el SDK de mapas de Google, por lo que cambiar el tipo de mapa se maneja pasando una serie de variables al objeto mapa que todo el rato mantenemos durante la ejecución de la aplicación.

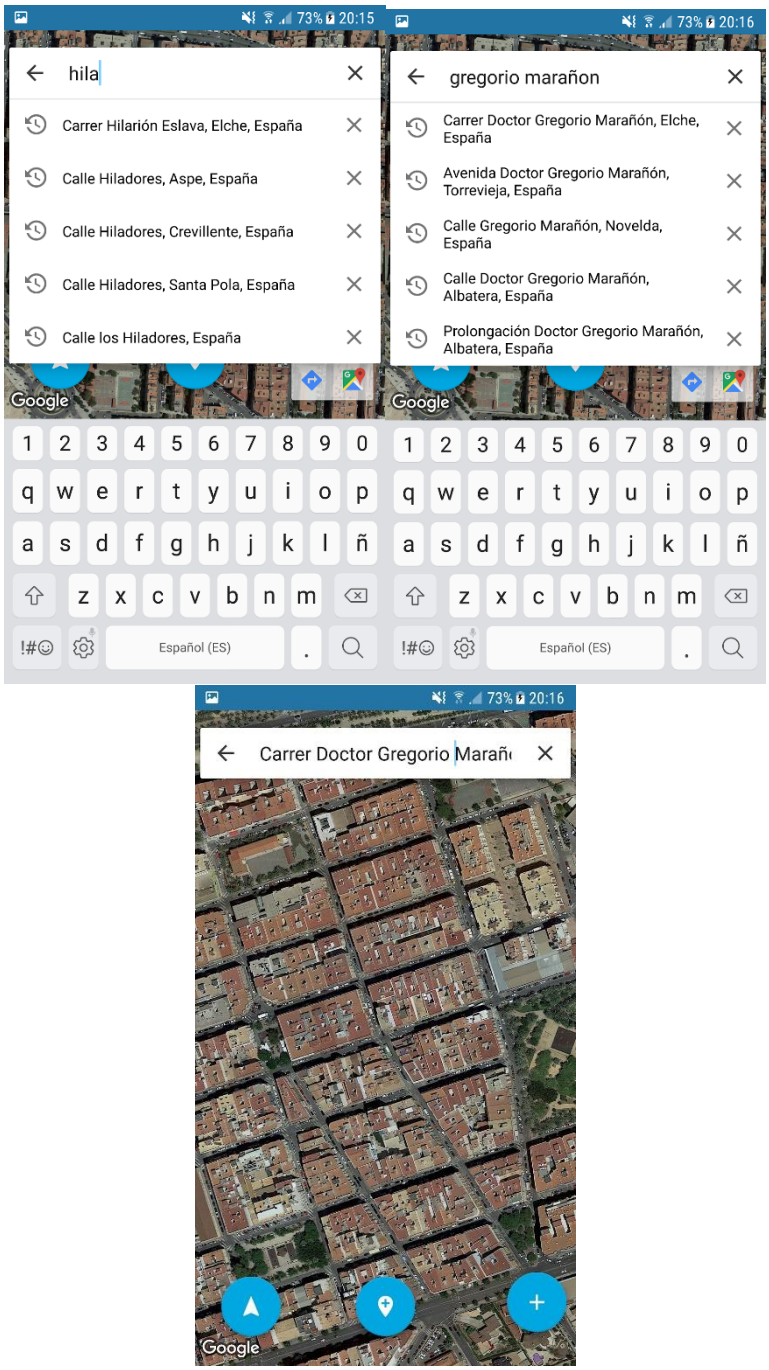

Figura 33 – Direccionamiento y búsqueda

Hemos hablado antes de esta barra tan útil para movernos por el mapa y cómo fue creada y en esta transición podemos ver cómo buscamos y al mismo tiempo que escribimos la barra de búsqueda posee una lista de sugerencias.
Además, si pulsamos en cualquiera de los resultados de la lista desplegada nos llevaría hasta allí sin problema. Esto ayuda mucho a buscar rápidamente lugares y direcciones.

Lo que hicimos con esta funcionalidad fue personalizarla para que cuando buscásemos una calle nos dirigiese, pero no crease ningún marker por dos motivos:

- Crear un marker en una zona con algunos que marcasen plazas de aparcamiento podría ser confuso.
- No veíamos útil crear este tipo de marcadores para una calle o un lugar cuando ya tenemos aplicaciones para ello en nuestros teléfonos.

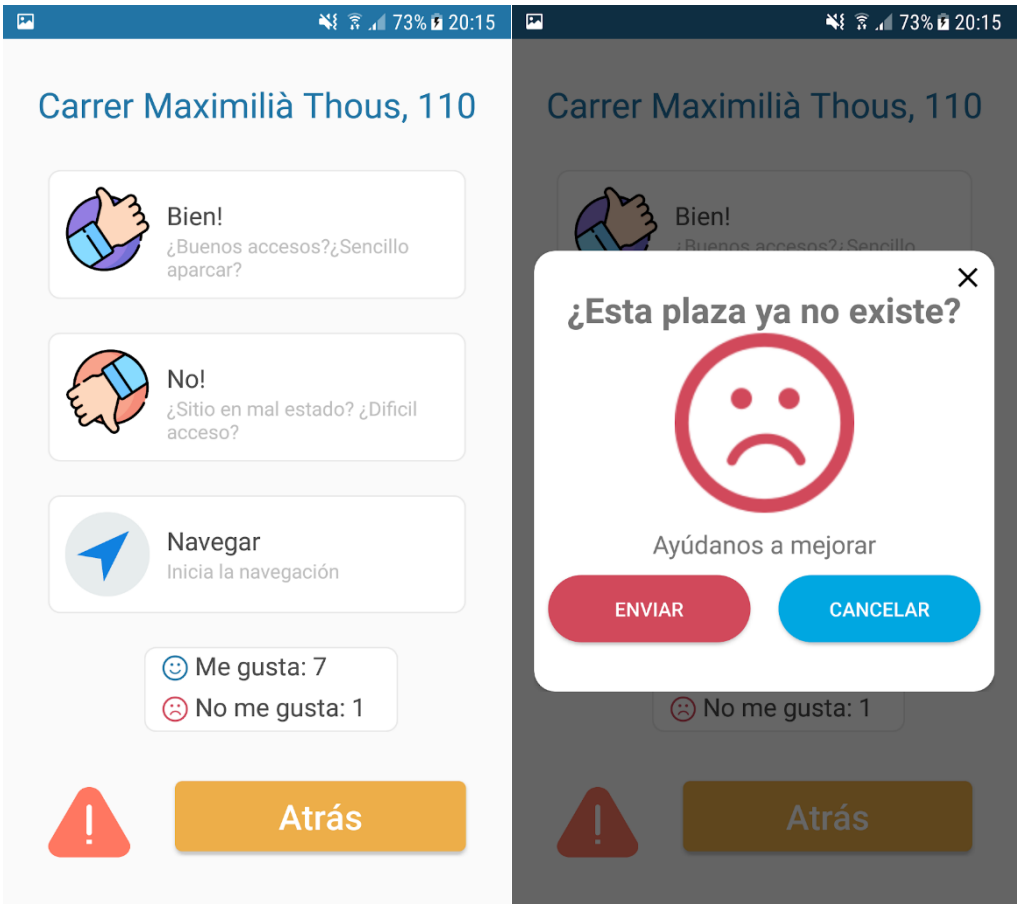

Figura 34 – Detalle e incidencia

Esta pantalla es la interfaz de detalle de una plaza de aparcamiento. Creamos esta actividad para poder tener funcionalidades y datos sobre una plaza de aparcamiento de manera aislada con respecto al mapa.

Con esto podríamos conseguir que la interfaz quedase más clara y más despejada sin un mapa por detrás que llenase más la vista del usuario.

En lo más alto tenemos el nombre de la calle en la que se encuentra la plaza, por otro lado, tenemos tres botones centrales. Con los dos primeros podemos votar si la plaza nos parece buena o no y el último sería para que se nos abriese Google maps y pudiésemos llegar con la navegación hasta la plaza.

En la parte baja de la pantalla tenemos dos botones. El rojo con la exclamación es para enviar una incidencia sobre la plaza. Esta funcionalidad no existía cuando hicimos el

análisis y diseñamos los mockups, pero vimos que podíamos implementarla a posteriori. Es una propiedad importante para nosotros, puesto que tener una plaza marcada en el mapa y que ya no existe es casi peor que no tenerla.

Ahora mismo solo podemos decir si la plaza ya no existe o cancelar, pero más adelante se podría implementar un sistema de sugerencias con mensajes por escrito o una lista de mensajes predeterminados para enviar. Y el botón atrás, que nos llevará de nuevo al mapa.

Como hemos visto en las anteriores figuras los resultados son muy parecidos a los deseados, aunque hemos cambiado cosas pensando en simplificar y ayudar a que las interfaces sean mejores.

Por un lado, se han tratado todos los puntos más importantes y aunque han surgido dudas y problemas técnicos al largo del desarrollo, no han sido demasiado preocupantes. Como por ejemplo cuando hubo que implementar un buscador con una lista de sugerencias con direcciones, el cual retrasó un poco el primer sprint.

Si bien es cierto que las horas que hemos podido dedicar al desarrollo no han podido ser muchas han sido suficientes como para quedar satisfecho con esta primera versión usable del producto. En lo personal quiero más.

Además, en un apartado posterior añadiremos las posibles mejoras y nuevas funcionalidades que se podrían añadir y a la vez no ser muy costosas de llevar a cabo. Todo esto para que la aplicación gane en riqueza y sea más completa, pero siempre buscando mantener unos de los objetivos principales: **la simplicidad**, usabilidad y accesibilidad.

10. Conclusiones y futuro del sistema

Para terminar este trabajo vamos a tratar este apartado en dos partes: por un lado, las posibles evoluciones que a las que puede optar una aplicación como EnableParking y por otro lado unas conclusiones generales y personales.

EnableParking ha sido un reto desde su inicio cuando nos reunimos para recabar información con la familia de AMFI los cuales tenían las ideas y la ilusión puesta en un sistema como este (necesario y útil), pasando por las valiosas entrevistas que tuvimos con Ana Puertas. Ana fue la responsable que llevó a cabo DisablePark y que luchó por que aquel proyecto pudiese ver la luz, y por la que acabamos poniendo el nombre actual a la aplicación que aquí finalizamos.

Sin embargo, el trabajo de EnableParking no acaba aquí puesto que es una aplicación y un sistema que merece un mimo especial. Ya que no solo es una aplicación que podamos descargar de nuestro store y podamos usar para consultar la tan preciada base de datos de plazas de aparcamiento para personas con problemas de movilidad reducida, sino que es un sistema en el que entra en juego un actor más. Esto es los dispositivos o balizas instaladas en las plazas de aparcamiento de los que se encargará nuestros compañeros de Artefactos de la Universidad de Alicante, de los cuales me siento un poquito parte.

Esto conseguirá darnos datos en tiempo real, lo cual vuelve a este proyecto más valioso si cabe.

Lo primero que se me ocurre proponer es analizar el coste de cambiar la aplicación a un lenguaje híbrido. Y manteniendo las ideas y la parte del backend conseguir una versión estable tanto para Android como para todas las plataformas posibles.

Además, algunas funcionalidades a nivel de software que se han quedado por el camino, principalmente por falta de tiempo, podría ser un login personalizado (ej. correo de Google) con el que trazar un perfil personalizado sobre las plazas más utilizadas. Con esto podríamos recibir notificaciones o ver un listado con el estado de cada plaza, sin buscarlo en el mapa, de forma que podamos consultar rápidamente cuáles están libres.

Otra línea de trabajo futura podría ser crear una funcionalidad mediante la cual los usuarios puedan escribir sobre esta plaza y en lugar de dar un simple me gusta/no me gusta puedan enviar su puntuación de una forma distinta (ej. estrellas) y escribir un comentario al respecto del estado de la plaza o el lugar en el que se encuentra. Esto podría dar información útil a las personas que se dispongan a utilizarlas posteriormente.

Esto junto con la opinión de los usuarios, a los cuales se les podría habilitar un buzón de sugerencias, serían las posibles hojas de ruta a seguir.

Pasando ya a las conclusiones generales, el objetivo de este proyecto era analizar, diseñar y reproducir una solución útil y sencilla para las personas que necesitan en su día a día utilizar las plazas de aparcamiento para personas con movilidad reducida e intentar ayudarles a que pierdan menos tiempo y esfuerzo mediante una herramienta en sus dispositivos móviles. Todo esto con los conocimientos adquiridos durante el Máster de

Desarrollo de Software para Dispositivos Móviles juntos con lo aprendido con anterioridad y durante el desarrollo del mismo, que ha sido mucho.

Con esto cubrimos unas necesidades no cubiertas ni por ayuntamientos, ni por empresas, solo por una estudiante con problemas de movilidad que consiguió que le creasen una aplicación por un montón de dinero y a la que luego dejaron de lado por falta de interés y de fondos.

Por otro lado, a nivel personal este trabajo de fin de máster ha sido duro. No por la aplicación o las tecnologías que, aunque no las conocía he aprendido mucho sobre ellas y sobre otras que sí que conocía he aprendido más, si no porque es un trabajo de larga duración y que me ha acompañado durante mucho tiempo en largas jornadas de trabajo que se volvían dobles al llegar a casa. Esto complicaba mucho el desarrollo, puesto que compaginar el trabajo y los estudios es una tarea dura, al menos para mí.

Pero ha sido enriquecedor, ya que he tenido más oportunidades de ser yo quien decida arquitectura, diseño, interfaces… y aunque me he equivocado muchas veces, he tenido la oportunidad de aprender de varios ámbitos de la programación que por desgracia no puedo en mi día a día. Y esto ha sido gracias a estos objetivos claros.

11. Referencias

1. Instituto nacional de estadística - http://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736176782&menu=resultados&secc=1254736194716&idp=1254735573175

2. AMFI: Asociación para la integración sociolaboral de personas con discapacidad física y sensorial - http://www.amfi.es/

3. ¿Por qué aparcar es una odisea para una persona con movilidad reducida? - http://www.predif.org/index.php?q=%C2%BFpor-qu%C3%A9-aparcar-es-una-odisea-par-una-persona-con-movilidad-reducida
   Las mil excusas de la inexcusable […] - https://www.bez.es/461629530/Aparcar-en-plazas-para-discapacitados.html

4. 2014. Noticia sobre las plazas para discapacitados - https://www.diarioinformacion.com/elche/2014/02/19/ayuntamiento-revisara-plazas-aparcamiento/1470936.html

5. Google - https://www.google.es/

6. Stores moviles - https://www.apple.com/es/ios/app-store/ - https://play.google.com/store

7. Disable Park - https://www.disabledpark.com/

8. VantageMobility (VMI) - https://www.vantagemobility.com/

9. Artículo en VMI - https://www.vantagemobility.com/blog/accessible-parking-mobile-applications

10. BlueBadgeParking - https://bluebadgeparking.com/

11. WheelMate - https://www.coloplast.com/products/bladder-bowel/wheelmate/

12. Parking Mobility - https://www.parkingmobility.com/

13. Lienzo Canvas explicado - https://innokabi.com/lienzo-lean-canvas-el-lienzo-de-los-emprendedores/

14. Trello. Trello es un software de administración de proyectos con interfaz web y con cliente para iOS y android para organizar proyectos. - https://trello.com/

15. Wiki estándar IEEE 830 - https://wikis.fdi.ucm.es/ELP/Especificaci%C3%B3n_de_Requisitos_Software_seg%C3%BAn_el_est%C3%A1ndar_IEEE_830

16. ESP32 - https://www.espressif.com/en/products/hardware/esp32/overview

17. “Una operación atómica es cuando haces un cambio que afecta a múltiples entidades de la base de datos al mismo tiempo” - https://docs.microsoft.com/es-es/biztalk/core/atomic-transactions

18. Understandind accessibility - https://material.io/design/usability/accessibility.html#

19. Google maps - https://www.google.es/maps

20. Mapas - https://www.apple.com/es/ios/maps/

21. Balsamiq. Balsamiq Mockups es una aplicación gráfica de diseño de interfaz de usuario y de creación de marcos de alambre para sitios web. - https://balsamiq.com/wireframes/

22. Material Design - https://material.io/design/

23. Coolors tool - https://coolors.co/

24. Repositorio de iconos - https://www.flaticon.com/

25. The most diverse collection of icons ever - https://thenounproject.com/

26. An icon font for use with Google Maps API and Google Places… - http://map-icons.com/

27. ¿Cómo funcionan y qué son las actividades en android? - https://developer.android.com/guide/components/activities.html?hl=ES

28. MaterialSearchbar en la que nos basamos - https://github.com/mancj/MaterialSearchBar

29. Cómo utilizar firestore - https://firebase.google.com/docs/firestore/query-data/listen?hl=es-419

30. Diagram tool - https://www.draw.io/

31. Cloud Firestore. Cloud Firestore es una base de datos flexible y escalable para la programación en servidores y dispositivos móviles - https://firebase.google.com/docs/firestore?hl=es

Anexo I:

Bibliografía, documentos y sitios web consultados, aunque no citados directamente:

Sunrise Medical. Solicitar una plaza de aparcamiento para discapacitados. ¿Qué hace falta? - https://www.sunrisemedical.es/blog/solicitar-plaza-aparcamiento-discapacitados

El periódico. Campaña para respetar las plazas para discapacitados - https://www.elperiodicodearagon.com/noticias/aragon/campana-respetar-plazas-discapacitados_1243314.html

Alejandra Otero. La DGT llama al civismo en su nueva campaña: no aparques en plazas para personas con discapacidad si no lo eres - https://www.motorpasion.com/seguridad/dgt-llama-al-civismo-su-nueva-campana-no-aparques-plazas-para-personas-discapacidad-no-eres

Bitbucket. Bitbucket es un servicio de alojamiento basado en web, para los proyectos que utilizan el sistema de control de versiones Mercurial y Git. - https://bitbucket.org

Firebase Database. Firebase Realtime Database es una base de datos alojada en la nube. - https://firebase.google.com/docs/database/?hl=es